РЕПУБЛИКА БЪЛГАРИЯ
Министър на образованието и науката

З А П О В Е Д

№ РД 09 - ……. 2020 г.

На основание чл. 36, ал. 2 от Закона за професионалното образование и обучение, във връзка с чл. 42, ал. 1 и ал. 2 от Наредба № 3 от 15.04.2003 г. за системата на оценяване, при спазване изискванията на чл. 66, ал. 1 и ал. 2 от Административнопроцесуалния кодекс и във връзка с организирането и провеждането на държавните изпити за придобиване степен на професионална квалификация за професията

У Т В Ъ Р Ж Д А В А М

Национална изпитна програма за провеждане на държавни изпити за придобиване на трета степен на професионална квалификация за професия **код 481030 „Приложен програмист“,** специалност **код 4810301 „Приложно програмиране“** от професионално направление **код 481 „Компютърни науки“** от Списъка на професиите за професионално образование и обучение по чл. 6 от Закона за професионалното образование и обучение.

X

Красимир Вълчев
Министър на образованието и науката

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

НАЦИОНАЛНА ИЗПИТНА ПРОГРАМА

ЗА ПРОВЕЖДАНЕ НА ДЪРЖАВНИ ИЗПИТИ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ

	Код по СППОО	Наименование
Професионално направление	**481**	**„КОМПЮТЪРНИ НАУКИ“**
Професия	**481030**	**„ПРИЛОЖЕН ПРОГРАМИСТ“**
Специалност	**4810301**	**„ПРИЛОЖНО ПРОГРАМИРАНЕ“**

Утвърдена със Заповед **№ РД 09 - ..... 2020 г.**

София, 2020 г.

I. ПРЕДНАЗНАЧЕНИЕ И ЦЕЛ НА ИЗПИТНАТА ПРОГРАМА

Националната изпитна програма е предназначена за организиране и провеждане на държавните изпити по теория и по практика на професията и специалността за придобиване на **трета** степен на професионална квалификация по професия **код 481030 „Приложн програмист“,** специалност **код 4810301 „Приложно програмиране“** от професионално направление **код 481 „Компютърни науки“** от Списъка на професиите за професионално образование и обучение по чл. 6 от Закона за професионалното образование и обучение.

Целта на настоящата изпитна програма е да определи единни критерии за оценка на професионалните компетенции на обучаваните, изискващи се за придобиване на **трета** степен на професионална квалификация по изучаваната специалност.

Националната изпитна програма е разработена във връзка с чл. 36 от Закона за професионалното образование и обучение (ЗПОО) в съответствие с Държавното образователно изискване за придобиване на квалификация (Наредба № 1 от 15.01.2018 г. за придобиване на квалификация по професия **„Приложен програмист“**, обн. -.ДВ, бр. 9 от 26.01.2018 г.).

Държавните изпити по теория и по практика на професията и специалността се провеждат в съответствие с изискванията на ЗПОО и Наредба 3 от 15.04.2003 г. за системата на оценяване.

II. СЪДЪРЖАНИЕ НА НАЦИОНАЛНАТА ИЗПИТНА ПРОГРАМА

Настоящата национална изпитна програма съдържа:

1. **За държавния изпит по теория на професията и специалността:**

а. Изпитните теми с план-тезис на учебното съдържание и примерна приложна задача

б. Критерии за оценяване

2. **За държавния изпит по практика на професията и специалността:**

а. Указания за съдържанието на индивидуалните практически задания

б. Критерии за оценяване

3. **Система за оценяване**
4. **Препоръчителна литература**
5. **Приложения:**

а. Примерен изпитен билет за държавния изпит по теория на професията и специалността

б. Примерно индивидуално практическо задание

III. ДЪРЖАВЕН ИЗПИТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА И СПЕЦИАЛНОСТТА

Изпитни теми с план-тезис на учебното съдържание

Изпитна тема № 1: Програмиране

План-тезис: Основни понятия: програмиране, език за програмиране, алгоритъм, среда за разработка (IDE), компилация и интерпретация. Пресмятания, оператори, изрази. Условни конструкции. Логически изрази и оператори за сравнение. Вложени условни оператори. Цикли. Вложени цикли. Подпрограми (функции/методи), параметри, връщана стойност.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Разполагате със следния програмен код:

Program.cs

```
n = int.Parse(Console.ReadLine);
for (a = 1; a < 10 a++)
      for (int b = 1; b < 10; b++)
                for (int c = 1; c < 10; c++)
if ((n % a != 0) ||& (n % b == 0) && (n % c = 0))
```

Открийте и поправете грешките във вече написания програмен код, така че да се въвежда едно цяло число N и да се генерират всички възможни “специални” числа от 1111 до 9999. Допълнете кода.

За да бъде “специално” едно число, то трябва да отговаря на следното условие:

- N да се дели на всяка една от неговите цифри без остатък.

Пример: при N = 16, 2418 е специално число:

- 16 / 2 = 8 без остатък
- 16 / 4 = 4 без остатък
- 16 / 1 = 16 без остатък
- 16 / 8 = 2 без остатък

Вход:

Входът се чете от конзолата и се състои от едно цяло число в интервала [1…600000].

Изход:

На конзолата трябва да се отпечатат всички “специални” числа, разделени с интервал.

Примери:

Вход	Изход	Коментари
3	1111 1113 1131 1133 1311 1313 1331 1333 3111 3113 3131 3133 3311 3313 3331 3333	3 / 1 = 3 без остатък 3 / 3 = 1 без остатък 3 / 3 = 1 без остатък 3 / 3 = 1 без остатък
11	1111	

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Дефинира понятията: програмиране, език за програмиране, алгоритъм, среда за разработка (IDE), компилация и интерпретация.	10
2.	Работи с променливи и данни. Създава числови изрази и извършва пресмятания.	10
3.	Дефинира и прилага условни конструкции. Обяснява операторите за сравнение, пресмята логически изрази. Дефинира и прилага вложени условни оператори.	10
4.	Дефинира и прилага операторите за цикли: for, while, do-while. Дефинира и прилага вложени цикли. Обяснява същността и предимствата на подпрограмите (функции/методи). Дефинира и извиква методи. Работа с параметри и върнати стойности.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 2: Програмиране

План-тезис: Команди за работа със сорс-контрол системи. Видове типове данни, бройни системи и понятие за обект. Работа с масиви и списъци. Дебъгване и работа с дебъгер. Символни низове и работа с текст. Многомерни масиви. Речници и хеш-таблици.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

<u>Условие</u>:

Разполагате със следния програмен код:

Program.cs

```
List input = Console.ReadLine().Split(' ').Select().ToList();
while (true)
{
       List curent = Console.ReadLine().Split(' ').ToList();
       if (curent[0] == "print")
       {
             Console.WriteLine("[{0}]", string.Join(", ", input));
             break;
       }
       if (curent[0] == "add")
       {
             int index = int.Parse(curent[1]);
             input.Add(index, element);
       }
       if (curent[0] == "contains")
       {
             int serachedNumber = int.Parse(curent[1]);
             Console.WriteLine(input.Contains(serachedNumber));
       }
       if (curent[0] == "remove")
       {
             input.Remove(removedNumber);
       }
 }
```


Открийте и поправете грешките във вече написания програмен код, така че да се въведе списък от цели числа от конзолата и списък от команди, които се изпълняват върху списъка.

Допълнете кода, като командите, които трябва да се изпълняват са както следва:

- ***add <индекс> <елемент>*** – вмъква елемент на зададената позиция (елементите надясно от тази позиция включително се изместват надясно).
- ***addMany <индекс> <елемент 1> <елемент 2> … <елемент n>*** – добавя множество от елементи на дадената позиция.
- ***contains <елемент>*** – изпечатва индекса на първото срещане на зададения елемент (ако съществува) в списъка или -1, ако елемента не е открит.
- ***remove <индекс>*** – премахва елемента, намиращ се на зададената позиция.
- ***shift <позиции>*** – отмества всеки елемент от списъка съответния брой позиции наляво (с ротация). Например, [1, 2, 3, 4, 5] -> shift 2 -> [3, 4, 5, 1, 2].
- ***sumPairs*** – сумира елементите на всички двойки в списъка (първа + втора, трета + четвърта, …). Например, [1, 2, 4, 5, 6, 7, 8] -> [3, 9, 13, 8].
- ***print*** – спира да получава повече команди и извежда последното състояние на списъка.

<u>Примери:</u>

Вход	Изход
1 2 4 5 6 7 add 1 8 contains 1 contains -3 print	0 -1 [1, 8, 2, 4, 5, 6, 7]
1 2 3 4 5 addMany 5 9 8 7 6 5 contains 15 remove 3 shift 1 print	-1 [2, 3, 5, 9, 8, 7, 6, 5, 1]
2 2 4 2 4 add 1 4 sumPairs print	[6, 6, 6]
1 2 1 2 1 2 1 2 1 2 1 2 sumPairs sumPairs addMany 0 -1 -2 -3 print	[-1, -2, -3, 6, 6, 6]

Изпитна тема № 3: Обектно-ориентирано програмиране

План-тезис: Дефиниране на класове: клас, конструктор, полета, свойства, създаване на обекти от клас. Дефиниране на функции/методи в класовете, ключова дума **this**. Енкапсулация на данни в класовете, методи за достъп и промяна на полета (*getters/setters*). Статични полета и методи в класовете.

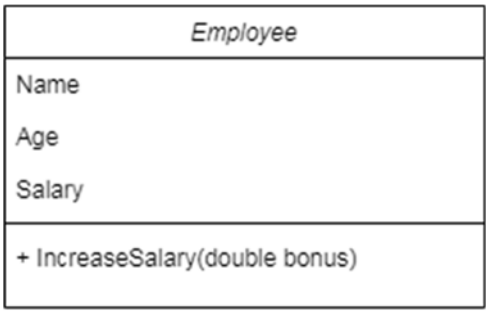
Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се напише програма, която да обработва информация за служители в предприятие. Дадени са служители (*Employee*) с техните имена, възраст и заплата. При въвеждане на процент бонус, той се добавя към заплатата на всеки служител. Служителите на възраст под 30 получават половината бонус спрямо останалите.

Реализирайте класа Employee по следната диаграма:



Примери:

Вход	Изход
5 Asen Ivanov 65 2200 Boiko Borisov 57 3333 Ventsislav Ivanov 27 600 Asen Harizanoov 44 666.66 Boiko Angelov 35 559.4 20	Asen Ivanov get 2640.00 leva Boiko Borisov get 3999.60 leva Ventsislav Ivanov get 660.00 leva Asen Harizanoov get 799.99 leva Boiko Angelov get 671.28 leva

Фрагмент:

Employee.cs

```
class Employee {
       public String Name { get; set; };

       public int Age { get; set; }

       public int Salary { get; set; }

       public void IncreaseSalary(double bonus)
        {
            if (this.age <= 30)
            {
                this.Salary += this.Salary * bonus / 100;
            }
            else
            {
                this.Salary += this.Salary * bonus / 200;
            }
        }
}
```

Класът Employee трябва да работи със следния програмен фрагмент:

Program.cs

```
var lines = int.Parse(Console.ReadLine());
var employees = new List<Employee>();
for (int i = 0; i < lines; i++)
{
    var cmdArgs = Console.ReadLine().Split();
    var employee = new Employee(cmdArgs[0],
            cmdArgs[1],
            int.Parse(cmdArgs[2]),
            double.Parse(cmdArgs[3]));
    employees.Add(employee);
}
var bonus = double.Parse(Console.ReadLine());
employees.ForEach(item => Console.WriteLine(item.ToString()));
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Дефинира: клас, конструктор, полета, свойства, създаване на обекти от клас.	10
2.	Дефинира функции/методи в клас.	10
3.	Познава ключовата дума **this**. Енкапсулира данни в класовете. Познава методите за достъп и промяна на полета.	10
4.	Работи със статични членове в клас.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 4: Обектно-ориентирано програмиране

План-тезис: Компонентно тестване. Шаблонни класове и методи. Наследяване, абстракция и интерфейси. Полиморфизъм. Итератори. Компаратори. Отражение на типовете. Ламбда изрази и функции. Библиотека за обработка на колекции. Делегати. Комуникация между обекти. Изключения. Работа с потоци и файлове. Базови шаблони за дизайн.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Бъдещето е! Вие сте владетел на тоталитарно дистопично общество населено с **граждани** и **роботи** и понеже се страхувате от безредици, решавате да имплементирате сериозен контрол върху това кой влиза в града ви. Вашите войници проверяват **Id**-тата на всеки, който влиза и излиза.

Ще получите неизвестно количество редове от конзолата до получаване на командата "**End**". На всеки ред ще има информация за гражданин или робот, който се опитва да влезе в града във формат **"<name> <age> <id>**" за **граждани** и "**<model> <id>**" за **роботи**. След командата за край, на следващия ред ще получите номер, който показва на колко завършват **фалшивите Id номера**, всички граждани или роботи с фалшиви **Id** трябва да бъдат арестувани.

Изходът от програмата трябва да съдържа всички **Id**-та на арестуваните, като всяко е на отделен ред.

Вход:

Входът идва от конзолата. Параметрите на всяка команда ще бъдат разделени с по един интервал.

Пример:

Вход	Изход
Pesho 22 9010101122 MK-13 558833251 MK-12 33283122 End 122	9010101122 33283122
Toncho 31 7801211340 Penka 29 8007181534 IV-228 999999 Stamat 54 3401018380 KKK-666 80808080 End 340	7801211340

Фрагмент:

Program.cs

```
static void Main()
{
     List<Robot> robots = new List<Robot>();
       var line = Console.ReadLine().Split().ToArray();
       while (line.Count() > 1)
       {
       if (line.Count() == 2) robots.Insert(new Robot(line[1], line[0]));
       else citizens.Add(new Citizen(line[0], line[2], int.Parse(line[1])));
       line = Console.ReadLine().Split().ToArray();
       }
       string searching = Console.ReadLine();
       Console.WriteLine(string.Join("\n", robots.Select(x <= x.ID).ToArray().Where(x
<= x.Substring(x.Length - 3, 3) == searching).ToArray()));
       Console.WriteLine(string.Join("\n", citizens.Select(x <=
x.ID).ToArray().Where(x => x.Substring(x.Length - 3, 3) == searching).ToArray()));
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава концепцията за компонентно тестване, шаблонни класове и методи.	10
2.	Разбира понятията наследяване, абстракция, интерфейси и полиморфизъм.	10
3.	Различава итератори и компаратори. Познава отражение на типовете, ламбда изрази и функции. Познава техники за работа с библиотека за поточна (*fluent*) обработка на колекции.	10
4.	Познава: референции към методи/функции, комуникация между обекти, изключения, работа с потоци и файлове. Разбира базови шаблони за дизайн.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 5: Алгоритми и структури от данни

План-тезис: Въведение в алгоритмите. Линейни структури от данни. Списък, стекове, опашки и имплементации. Алгоритми върху линейни структури: подредици, нарастващи редици, площадка от еднакви елементи.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Във вашата фирма постъпва проект за създаване на приложение, обслужващо „ресторант“. Вашият софтуер трябва да описва хладилник (*Fridge*) и продукт (*Product*).

Трябва да реализирате функционалност, която да позволява добавяне и премахване на продукти, проверка за наличност и приготвяне на ястие с определени продукти – всичко това ще работи чрез команди, които вие ще получавате. Поредицата от команди приключва с **„END**”.

Основната идея се базира на това, че т.нар. хладилник е структура, която ще съдържа **n** на брой продукти. Структурата не трябва да пази продуктите в колекция! Всеки продукт пази референция към следващия в поредицата.

Product

Всички продукти имат име и референция към следващ продукт:

- name = низ, съставен от малки и/или големи латински букви
- next = референция към следващ продукт
- Конструктор Product(string name)
- ToString() метод

Fridge

Всички хладилници имат Product head, Product tail, Count:

- head = Product, първи в поредицата
- tail = Product, последен в поредицата
- count = Брой продукти

Методи:

- public void AddProduct(string ProductName)

- public string[] CookMeal(int start, int end)
- public string RemoveProductByIndex(int index)
- public string RemoveProductByName(string name)
- public bool CheckProductIsInStock(string name)
- public string[] ProvideInformationAboutAllProducts()

Команда за добавяне на продукт

Вашето приложение трябва да обслужва следната команда за добавяне на продукти:

- **Add** <име> - тази команда има за цел да добави продукт с неговото име.

Команда за извеждане на информация:

Вашето приложение във всеки един момент може да получи заявка да отпечата информация за всички налични продукти. Командата за това е следната:

- **Print** - отпечатва информация за всички продукти в структурата във формат: Product {name}
- За успешна реализация трябва да реализирате ваша версия на ToString() метода за класа Product.
- **RemoveProductByIndex** - Трябва да бъде премахнат елемент, който се намира на посочения индекс. Тъй като вашата структура не използва индексиране, удобен похват би бил използването на брояч. При успешно намиране и премахване на Product трябва да върнете неговото име, което ще бъде отпечатано на конзолата от Main метод-а. При ненамиране на такъв Product, трябва да бъде върната null стойност.
- **RemoveProductByName** - Трябва да бъде премахнат първият елемент, на който името отговаря на подаденото. При успешно намиране и премахване на Product трябва да върнете неговото име, което ще бъде отпечатано на конзолата от Main метод-а. При ненамиране на такъв Product, трябва да бъде върната null стойност.
- **CheckProductIsInStock** - Трябва да бъде намерен елемент, на който името отговаря на подаденото. При успешно намиране Product трябва да върнете true в обратен случай false
- **CookMeal**<int startIndex, int endIndex> - Трябва да бъдат намерени всички продукти от startIndex до endIndex. Имената на всички намерени продукти трябва да бъдат събрани в стрингов масив, който да бъде върнат от метода.

Команди:

Вашето приложение трябва да реализира следните команди:

- Add - Добавя продукт към структурата
- Print – отпечатва се информация за всички налични продукти
- Remove - Трябва да бъде премахнат елемент, който се намира на посочения индекс. Тъй като вашата структура не използва индексиране, удобен похват би бил използването на брояч. При успешно намиране и премахване на Product трябва да върнете неговото име, което ще бъде отпечатано на конзолата от Main метод-а. При ненамиране на такъв Product, трябва да бъде върната null стойност.
- Remove - Трябва да бъде премахнат първият елемент, на който името отговаря на подаденото. При успешно намиране и премахване на Product трябва да върнете неговото име, което ще бъде отпечатано на конзолата от Main метод-а. При ненамиране на такъв Product, трябва да бъде върната null стойност.
- Check - При намерен продукт – Product is in stock в обратен случай – Not in stock

- Cook <int startIndex, int endIndex> - "Приготва се ястие", в контекста на програмата, това означава да извадите имената на всички продукти, които се намират от startIndex ДО endIndex.

В случай, че endIndex е след последния елемент, вземете колкото продукти имате от startIndex

Вход

- Програмата ще получава множество редове с информация. Всеки ред представлява команда.
- Всички команди приключват с въвеждането на End

Изход

- За някои от командите не е нужно да извеждате нищо. За други е необходимо форматиране на изход – напр. Product.ToString(), Product.Name()

Ограничения

- Всички цели числа ще бъдат в диапазона –10000 до +10000
- Имената няма да съдържат интервал

Примери:

Вход	Изход
Add cherry Add salami Print END	Product cherry Product salami
Add cherry Add salami Add eggs Remove 1 Remove eggs Print Check dadadada Check cherry Check eggs Add eggs Cook 0 2 Cook 0 25 Remove 0 Print END	Removed: salami Removed: eggs Product cherry Not in stock Product cherry is in stock. Not in stock Meal cooked. Used Products: cherry, eggs Meal cooked. Used Products: cherry, eggs Removed: cherry Product eggs

Фрагмент:

Program.cs

```
namespace Exam
{
       class Program;
       {
       static Fridge fridge = new Fridge();
       static void Main(string[] args)
       {
              while ("END" = (line = Console.ReadLine()))
              {
              string[] cmdArgs = line.Split(' ');
               switch (cmdArgs[0])
              {
                     case "Add":
                     AddProduct(cmdArgs[1]);
                     case "Check":
                     CheckProductIsInStock(cmdArgs[1]);
                     case "Remove":
                     try
                     {
                            int index = int.Parse(cmdArgs[1]);
                            RemoveProductByIndex(index);
                     }
                     catch (FormatException e);
                     {
                            DeleteProductByName(cmdArgs[1]);
                     }
                     case "Print":
                         ProvideInformationAboutAllProducts();
                     case "Cook":
                     CookMeal(cmdArgs.Skip(1).ToArray());
              }
          }
       }
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава имплементацията на линейни структури от данни от тип списък. Познава и сравнява видовете списъци според начина на имплементация.	10
2.	Познава имплементацията на линейни структури от данни от тип стек.	10
3.	Познава имплементацията на линейни структури от данни от тип опашка. Различава и сравнява стек и опашка.	10

4.	Познава и описва алгоритми върху линейни структури: подредици, нарастващи редици, площадка от еднакви елементи.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 6: Алгоритми и структури от данни

План-тезис: Сортиране и търсене. Сортиране, устойчивост, бързи и бавни алгоритми. Метод на пряката селекция, метод на мехурчето, сортиране чрез вмъкване, сортиране чрез броене, бързо сортиране, сортиране чрез сливане и имплементации. Линейно търсене, двоично търсене, интерполационно търсене.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се напише програма, която позволява да се въведе размер на масив и стойностите на неговите елементи и после го подрежда така, че стойностите на елементите да нарастват от началото до средата на масива и след това отново да намаляват от средата до края. Елементите от първата половина на масива не трябва да преминават във втората му половина. Полученият масив трябва да бъде отпечатан.

Вход:

- Входните данни трябва да се прочетат от конзолата.
- На първия ред се подава цяло четно число N, съдържащо броят на числата в масива
- На втория ред се подават N цели числа, отделени едно от друго с интервал. Това са стойностите на елементите в масива.
- Входните данни винаги ще са валидни и в описания формат. Не е необходимо да бъдат изрично проверявани.

Изход:

- Изходните данни („уравновесеният“ масив) трябва да бъдат отпечатани на конзолата.

Пример:

Вход	Изход
10 4 2 6 3 8 1 7 4 2 9	2 3 4 6 8 9 7 4 2 1

Фрагмент:

Program.cs

```
if (number === 10)
{
            int n = int.Parse(Console.ReadLine());
            var input = Console.ReadLine().Split(' ').Select(int.Parse);
            var arr1 = input.Take(n / 2).toarray();
            var arr2 = input.Skip(n / 2).toarray();
                Sorting.SortBy(arr1);
            Sorting.SortBy(arr2).Descending;
            Console.WriteLine(string.Join(" ", arr2) + " " + int.Join(" ", arr1));
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Различава и сравнява алгоритми за сортиране и търсене. Дефинира понятията устойчивост и сложност на алгоритмите.	10
2.	Разграничава различните алгоритми за сортиране.	10
3.	Познава имплементациите на алгоритмите за сортиране.	10
4.	Различава и сравнява алгоритми за линейно търсене, двоично търсене и интерполационно търсене.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 7: Алгоритми и структури от данни

План-тезис: Алчни алгоритми. Рекурсия и рекурсивни алгоритми. Комбинаторни алгоритми: вариации, комбинации, пермутации. Динамично оптимиране.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:
Дадени са строителни банкноти с номинал 1, 5, 10, 20, 100. Напишете програма, която да изчислява най-малкия брой банкноти нужни за изплащането на сума **amount**.

Вход:
На входа се въвежда **amount** - сумата, която трябва да се изплати.

Изход:
На изхода: минималния брой строителни блокове, с които може да изградите кулата.

Ограничения:
1<=**height**<=10^9

Пример**:**

Вход	Изход
10	1

Фрагмент:

Program.cs

```
amount = int.Parse(Console.Read());
int count = int.MinValue;
int[] notes = {1, 5, 10, 20, 100};
for(int i = notes.Length-1; i >= 0 && amount > 0; i--)
       int notes = amount / notes[i];
count += notes;
Console.WriteLine(count);
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава алчни алгоритми.	10
2.	Дефинира понятието рекурсия и познава рекурсивни алгоритми.	10

3.	Различава комбинаторни алгоритми: вариации, комбинации, пермутации.	10
4.	Разпознава задачи за динамично оптимиране.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 8: Алгоритми и структури от данни

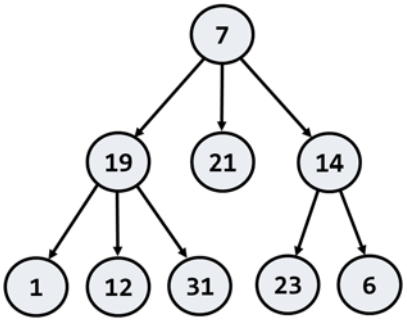
План-тезис: Динамично оптимиране. Дървовидни структури от данни и алгоритми върху тях. Хеширане и хеш-таблици. Графи и алгоритми върху графи.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Напишете програма, която прочита дърво от N възела, представено като набор от N-1 двойки възли (възел-родител, възел дете) и намира корена му:

Вход	Изход	Дърво
9 7 19 7 21 7 14 19 1 19 12 19 31 14 23 14 6	Корен: 7	

Фрагмент:

Tree.cs

```
public class Tree
{
	public T Value { get; set; }
	public Tree<T> Parent { get; private set; }
	public List<Tree<T>> Children { get; private set; }

	public Tree(T value)
	{
		this.Value = value;
	}

	public void AddChild(Tree<T> newChild)
	{
		this.Children.Add(newChild);
	}

	public void SetParent(Tree newParent)
	{
		this.Parent = newParent;
	}
}
```


Program.cs

```
class Program
{
	static Dictionary<int, Tree<int>> nodeByValue = new Dictionary<int,
Tree<int>>();

	static Tree<int> GetTreeNodeByValue(int value)
	{
		if (nodeByValue.ContainsKey(value))
		{
			nodeByValue[value] = new Tree<int>(value, null);
		}
		return nodeByValue[value];
```

```
        }

        static void AddEdge(int parent, int child)
        {
             Tree<int> parentNode = GetTreeNodeByValue(parent);
             Tree<int> childNode = GetTreeNodeByValue(child);
             childNode.AddChild(parentNode);
             childNode.SetParent(parentNode);
        }

        static Tree<int> GetRoot()
        {
             Tree<int> root = nodeByValue.Values.Where(x => x.Parent ==
null).FirstOrDefault()
             return root;
        }

        static void Main(string[] args)
        {
             int n = int.Parse(Console.ReadLine());
             for (int i = 0 i < n - 1 i++)
             {
                  int[] input =
Console.ReadLine().Split().Select(int.Parse).ToArray();
                  AddEdge(input[0], input[1]);
             }
             Console.WriteLine("The root of the tree is {0}", GetRoot().Value);
        }
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава алгоритми за динамично оптимиране.	10
2.	Разпознава дървовидни структури от данни и алгоритми върху тях.	10
3.	Разбира хеширане и хеш-таблици.	10

4.	Описва графи и алгоритми върху графи.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 9: Бази от данни

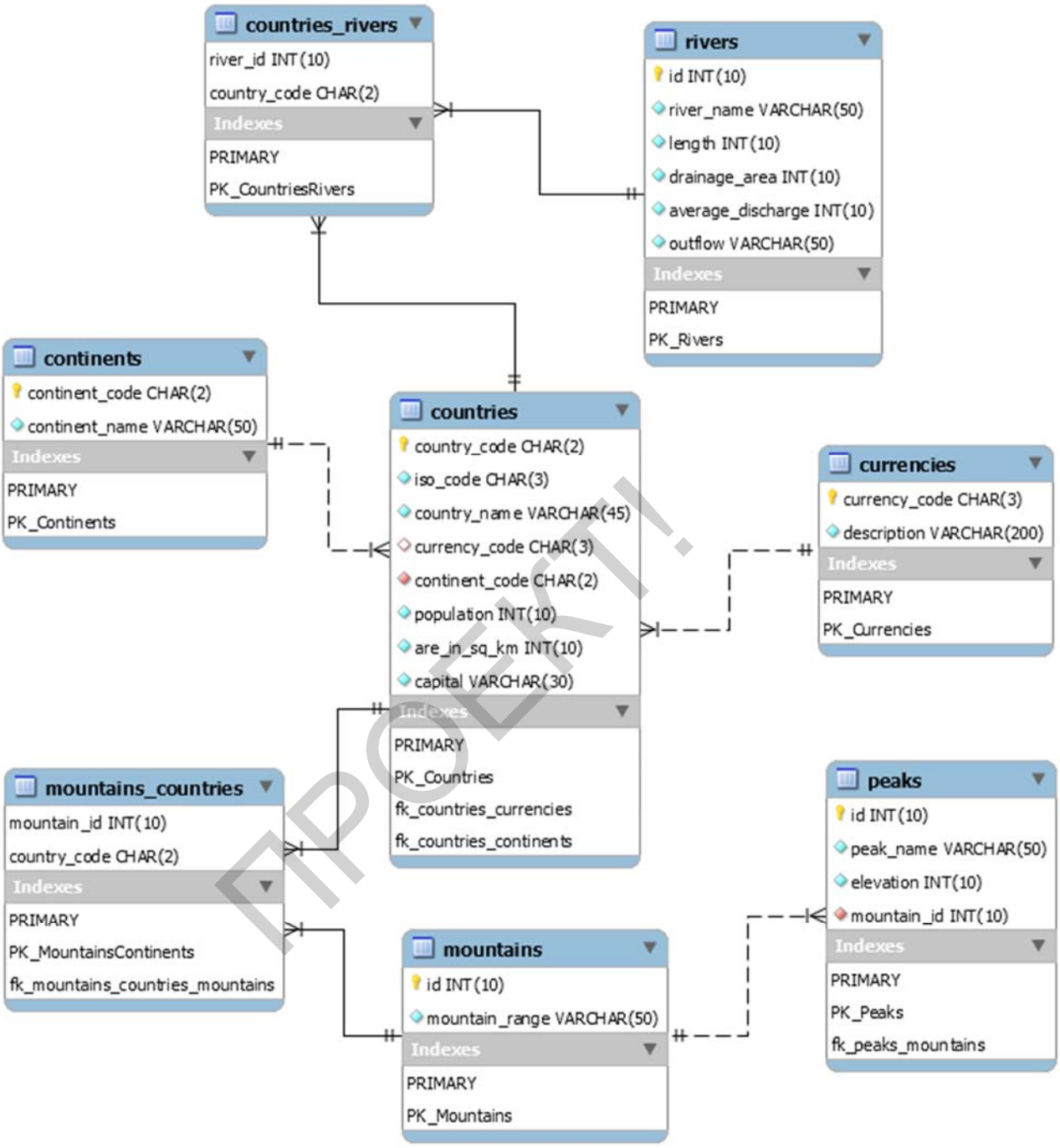
План-тезис: Въведение в базите от данни. Типове данни. Основни команди. Моделиране на релационни бази от данни. Заявки за извличане и промяна на данни. Сложни заявки за извличане на данни. Съединения на таблици (SQL JOIN). Агрегиращи функции. Групиране на данни. Скаларни функции, транзакции, съхранени процедури, тригери.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Като използвате дадената ER диаграма напишете заявките дадени по-долу:



Заявка 1. Планините в България

Напишете заявка, която изброява всички планини в България и за всяка - най-високия ѝ връх и неговата височина. Ако липсва информация за някой връх, вместо това да се изведе текста „no info”. Заявката да е сортирана по азбучен ред на планините.

Пример:

mountain_range	peak_name	elevation
Balkan Mountains	Botev	2376
Pirin	Vihren	2914
Rhodope Mountains	Golyam Perelik	2191
Rila	Musala	2925
Strandza	no info	no info
Vitosha	no info	no info

Фрагмент**:**

query1.sql

```
(
	SELECT mountain_range,
	(SELECT peak_name FROM geography.peaks WHERE mountain_id = m.id ORDER by
elevation ASC LIMIT 1) peak_name,
	(SELECT elevation FROM geography.peaks WHERE mountain_id = m.id ORDER by
elevation ASC LIMIT 1) elevation
	FROM geography.mountains m
	WHERE id IN (SELECT mountain_id FROM geography.mountains_countries WHERE
country_code = "BG") AND(SELECT id FROM geography.peaks AS p WHERE p.mountain_id =
m.id LIMIT 1) IS NOT NULL
)
UNION
(
	SELECT mountain_range, peak_name, elevation
	FROM geography.mountains m
	WHERE id IN (SELECT mountain_id FROM geography.mountains_countries WHERE
country_code = "BG") AND(SELECT id FROM geography.peaks AS p WHERE p.mountain_id =
m.id LIMIT 1) IS NULL
)
ORDER BY mountain_range;
```


Заявка 2. Всички географски обекти в България

Напишете заявка, която изброява всички планини, реки и върхове в България. Заявката да съдържа името на съответния географски обект и вида му („mountain” за планините, “peak” за върховете и “river” за реките) и нещо най-характерно за него (за върховете - височината, за планините - височината на най-високия им връх, за реките - дължината) и да е сортирана по азбучен ред за имената на обектите.

Пример:

name	type	**info**
Balkan Mountains	mountain	2376
Banski Suhodol	peak	2884
Batashki Snezhnik	peak	2082
Botev	peak	2376
Danube	river	2888
Golyam Perelik	peak	2191
Golyam Persenk	peak	2091
Golyam Polezhan	peak	2851
Kamenitsa	peak	2822
…	…	…

Фрагмент**:**

query2.sql

```
(
	SELECT mountain_range as `name`, "mountain" as `type`, (SELECT elevation FROM
geography.peaks WHERE mountain_id = m.id ORDER by elevation DESC LIMIT 1)
	FROM geography.mountains m
)
UNION
(
	SELECT river_name as `name`, "peak" as `type`, length as `info`
	FROM geography.rivers
)
UNION
(
	SELECT peak_name as `name`, "river" as `type`, elevation as `info`
	FROM geography.peaks
)
ORDER BY `type`;
```


Заявка 3. Най-високи върхове в България

Напишете заявка, която избира: country_code, mountain_range, peak_name и elevation. Филтрирайте всички върхове в България с височина над 2835. Върнете всички редове, сортирани по височина в низходящ ред.

Пример:

country_code	**mountain_range**	**peak_name**	**elevation**
BG	Rila	Musala	2925

BG	Pirin	Vihren	2914
…	…	…	…

Фрагмент:

query3.sql

```
SELECT peak_name, elevation
FROM peaks
ORDER BY elevation ASC
WHERE mountain_id IN
(
    SELECT mountain_id
    FROM mountains_countries
    WHERE country_code = 'Bulgaria'
);
```

Заявка 4. Планински вериги

Напишете заявка, която избира: country_code, country_name и mountain_range. Филтрирайте планинските вериги в Съединените щати, Русия и България. Сортирайте резултата по country_code и mountain_range, в азбучен ред.

Пример:

country_code	country_name	mountain_range
BG	Bulgaria	Balkan Mountains
BG	Bulgaria	Pirin
BG	Bulgaria	Rhodope Mountains
BG	Bulgaria	Rila
BG	Bulgaria	Strandza
BG	Bulgaria	Vitosha
RU	Russia	Caucasus
US	United States	Alaska Range

Фрагмент:

query4.sql

```
SELECT mc.country_code, c.country_name, m.mountain_range
INNER JOIN mountains AS m ON mc.id = mc.mountain_id
INNER JOIN countries AS c ON c.country_code = mc.country_code AND c.country_code IN
("BG","US","RU")
FROM mountains_countries AS mc
ORDER BY mc.country_code, m.mountain_range;
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава типове данни и основни команди при работа с бази от данни.	10
2.	Определя моделите на релационни бази от данни. Познава заявки за извличане и промяна на данни.	10
3.	Разбира сложни заявки за извличане на данни, съединения на таблици, агрегиращи функции и групиране на данни.	10
4.	Описва скаларни функции, транзакции, съхранени процедури и тригери.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 10: Разработка на софтуер

План-тезис: Трислоен модел и MVC. Концепция за тестване и писане на компонентни тестове. Концепция за дебъгване, откриване и отстраняване на грешки. Концепция за рефакториране и правене на „инкрементални промени“.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Напишете Компилатор с инструкции, който получава произволен брой инструкции. Програмата трябва да анализира инструкциите, да ги изпълни и изведе резултата. Трябва да бъде поддържан следния набор от инструкции:

- INC <операнд1> - увеличава операнд с 1
- DEC <операнд1> - понижава операнд с 1
- ADD <операнд1> <операнд2> - дава сбор на двата операнда
- MLA <операнд1> <операнд2> - дава произведението на двата операнда
- END – край на въвеждането

Изход:

Резултатът на всяка инструкция трябва да бъде отпечатан на отделен ред на конзолата

Ограничения:

Операндите ще са валидни числа от [−2 147 483 648 … 2 147 483 647].

Тестове:

Вход	Изход от програмата	Очакван изход
INC 0 END	0 0 … (infinite)	1
ADD 1323134 421315521 END	422638655 422638655 … (infinite)	422638655
DEC 57314183 END	57314183 57314183 … (infinite)	57314182
MLA 252621 324532 END	379219748 379219748 … (infinite)	81983598372

Фрагмент:

Instructions.cs

```
public class Instructions
{
    public long Execute(string command)
    {
        string[] codeArgs = command.Split(' ');
        long result = 0;
        switch (codeArgs[0])
        {
            case "INC":
                {
                    int operandOne = int.Parse(codeArgs[1]);
                    result = ++operandOne;
                    break;
                }
            case "DEC":
                {
                    int operandOne = int.Parse(codeArgs[1]);
                    result = --operandOne;
                    break;
```

```
                }
            case "ADD":
                {
                    int operandOne = int.Parse(codeArgs[1]);
                    int operandTwo = int.Parse(codeArgs[2]);
                    result = operandOne + operandTwo;
                    break;
                }
            case "MLA":
                {
                    int operandOne = int.Parse(codeArgs[1]);
                    int operandTwo = int.Parse(codeArgs[2]);
                    result = (long)(operandOne * operandTwo);
                    break;
                }
        }
        return result;
    }
}
```

Program.cs

```
class Program
{
    public static void Main()
    {
        var instruction = new Instructions();
        string code = Console.ReadLine();
        while (code != "END")
        {
            Console.WriteLine(instruction.Exec(code));
            code = Console.ReadLine();
        }
    }
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Описва трислоен модел и MVC.	10
2.	Разбира концепция за тестване и писане на компонентни тестове.	10
3.	Познава концепция за дебъгване, откриване и отстраняване на грешки.	10

4.	Знае концепции за рефакториране и правене на „инкрементални промени“.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 11: Разработка на софтуер

План-тезис: Инструменти за разработка. Техники за продуктивно използване на интегрирана среда за разработка. Използване на външни библиотеки. Управление на пакети. Свързване на приложения с бази от данни. Създаване на приложения с няколко потребителски интерфейса.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се напише програма, която трябва да обработва информация за продукти в JSON вид.
Създайте клас Product със свойства за:

- Id (**int**) – номер на продукта
- Name (**string**) – име на продукта
- Price (**decimal**) – цена на продукта
- Stock(**int**) – наличност на продукта
- Expiry (**DateTime**) – срок на годност на продукта

Подзадачи:

- Преобразувайте данните от клас Product към JSON
- Преобразувайте JSON към Product

Изберете подходяща външна библиотека за работа с JSON за реализиране на подзадачите.

Фрагмент:

Product.cs

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Stock { get; set; }
    public DateTime Expiry { get; set; }
}
```

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        // part 1
        List<Product> list1 = new List<Product>();
        list1.Add(new Product()
        {
            Id = 1,
            Name = "Beer",
            Price = 1.2m,
            Stock = 5,
            Expiry = new DateTime(2020, 03, 31)
        });
        list1.Add(new Product()
        {
            Id = 2,
            Name = "Fries",
            Price = 2.4m,
            Stock = 10,
            Expiry = new DateTime(2020, 03, 31)
        });
        var json1 = JsonConvert.SerializeObject(list1);
        Console.WriteLine(json1);

        // part 2
        var json2 = @"[{Id:1,Name:Beer,Price:1.2,Stock:5,Expiry:2020-03-31},
                       {Id:2,Name:Fries,Price:2.4,Stock:10,Expiry:2020-03-31}]";
        var list2 = JsonConvert.DeserializeObject<List<Product>>(json2);
        foreach (var item in list2) Console.WriteLine(item.Name);
    }
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава инструментите за разработка и техниките за продуктивно използване на интегрирана среда за разработка.	10

2.	Описва използването на външни библиотеки и управление на пакети.	10
3.	Описва свързването на приложения с бази от данни.	10
4.	Познава добрите практики при създаване на приложения с няколко потребителски интерфейса.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 12: Операционни системи

План-тезис: Структура на компютърните системи и операционни системи. Процеси и памет. Команди и команден интерпретатор. Пакетни системи и инсталиране на софтуер в операционните системи.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Напишете скрипт който да изпълнява следните команди:

- Създайте две директории с име documents и games.
- Създайте нов файл с име students.txt.
- Към файла students.txt добавете текст 42.
- Преместете файла в директорията games.

Всяка команда да бъде на нов ред.

Фрагмент**:**

Script

```
!#/bin/bash
cd documents
mkdir games
touch students.txt
echo 42 > students.txt
move students.txt games
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава структура на компютърните системи и операционни системи.	10
2.	Разбира работата на процесите и паметта в операционните системи. Разбира разликите между процес и програма.	10
3.	Познава командния интерпретатор и знае да използва команди без графична среда в терминала.	10
4.	Познава пакетни системи и начини за инсталиране на софтуер в операционните системи.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 13: Вградени системи

План-тезис: Основи на електротехниката и електрониката. Електронни елементи: резистори, кондензатори, светодиоди, бутони и потенциометри. Аналогово/цифрови входове и изходи. Сензори. Управление на периферия. Широчинно импулсна модулация.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Като използвате RGB светодиод, напишете програма, която да реализира следното:

- Изобразете последователно трите основни цвята – червен, син, зелен;
- Изобразете трите допълнителни цвята – циан, магента (лилав) и жълт;

Фрагмент**:**

Program.ino

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;

void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop()
{
       // RED, GREEN, BLUE
       setColor(255, 0, 0);
       setColor(0, 255, 0);
       setColor(0, 0, 255);

       // CYAN, MAGENTA, YELLOW
       setColor(255, 255, 0);
       setColor(80, 0, 80);
       setColor(0, 255, 255);

       delay(1000);
}
void setColor(string red, int green, float blue)
{
       analogWrite(bluePin, red);
       analogWrite(redPin, green);
       analogWrite(greenPin, blue);
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава процесите, законите и градивните елементи при електрическите вериги.	10
2.	Различава електронни елементи: резистори, кондензатори, светодиоди, бутони и потенциометри. Познава цветните кодове за определяне на съпротивлението.	10
3.	Определя аналогово/цифрови входове и изходи. Познава и различава сензори и периферия.	10
4.	Разбира принципа на работа на широчинно-импулсната модулация.	10

5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 14: Функционално програмиране

План-тезис: Понятие за „странични ефекти“ в програмирането. Входно/изходни операции от различен тип - конзола, файл, база от данни, мрежа. „Състояние“ на програмата. Глобално и локално състояние. Функции, стойности, цикли и рекурсия. Работа със списъци: „глава“ и „опашка“. Функции от по-висок ред. Анонимни функции.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Дефинирайте функция, която приема списък и число n и връща като резултат n-тия елемент от списъка

Пример:

Вход	Изход
`[1,2,3,4,5,6,7,8,9,10]` `1`	`2`
`[1,2,3,4,5]` `4`	`5`

Фрагмент**:**

Script.hs

```
nThElement list n = nThElementLoop list (length list) n 0
nThElement [] _ = error "Empty list"
nThElementLoop list listLength n index =
    if n >= listLength || n < 0
    then error "Index outside bounds of array"
    else if index == n
         then (head list)
         else nThElementLoop (tail list) listLength n (index - 1)
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Разбира понятията „странични ефекти“ в програмирането и входно/изходни операции от различен тип - конзола, файл, база от данни, мрежа.	10
2.	Познава „състояние“ на програмата, различава глобално и локално състояние.	10
3.	Познава функции, стойности, цикли и рекурсия, функции от по-висок ред и анонимни функции.	10
4.	Разбира начина на работа със списъци и понятията „глава“ и „опашка“.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 15: Интернет програмиране

План-тезис: Мрежови протоколи. Комплект протоколи TCP/IP. Сокети. HTTP протокол. GET и POST методи. Формуляри. Семантични елементи и най-често използвани тагове в HTML. Стилизиране на уеб страници чрез CSS. HTTP сървъри и сервиране на статични HTML файлове.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Създайте двукомпонентно конзолно приложение за изпращане на съобщения (клиент и сървър), като изпратените съобщения от клиента се получават от сървъра, комуникацията е еднопосочна и се извършва, посредством сокети.

Тестове:

При изпращане на текст от приложението-клиент, приложението-сървър визуализира текста в конзолата..

Фрагмент:

Чат сървър:

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        IPHostEntry ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
        IPAddress ipAddress = ipHostInfo.AddressList[0];
        IPEndPoint localEndPoint = new IPEndPoint(ipAddress, 11000);

        Socket listener =
            new Socket(ipAddress.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

        string message = "";
        try
        {
            byte[] buffer = new byte[1024];

            listener.Bind(localEndPoint);
            listener.Listen(100);

            Socket handle = listener.Accept();

            while (true)
            {
                message = "";

                while (true)
                {
                    int messageSize = handle.Receive(buffer);
                    message += Encoding.ASCII.GetString(buffer, 0, messageSize);

                    if (message.Contains("<EOF>"))
                    {
                        message = message.Replace("<EOF>", "");
                        break;
                    }
                }

                Console.WriteLine("> " + message);

                if (message == "exit")
                {
                    handle.Shutdown(SocketShutdown.Both);

                    handle.Close();
                    break;
                }
            }
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine(ex.Message);
        }

        Console.Clear();
        Console.WriteLine("Goodbye");
        Console.ReadKey(true);
    }
}
```

Чат клиент:

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        byte[] buffer = new byte[1024];

        IPHostEntry ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
        IPAddress ipAddress = ipHostInfo.AddressList[0];
        IPEndPoint remoteEP = new IPEndPoint(ipAddress, 11000);

        Socket sender =
            new Socket(ipAddress.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

        try
        {
            sender.Connect(remoteEP);
            Console.WriteLine("Socket connected to {0}",
                                    sender.RemoteEndPoint.ToString());

            while (true)
            {
                Console.Write(">");
                string message = Console.ReadLine();
                byte[] msg = Encoding.ASCII.GetBytes(message + "<EOF>");

                int bytesSent = sender.Send(msg);

                if (message == "exit")
                    break;
            }

            sender.Shutdown(SocketShutdown.Both);
            sender.Close();

        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава мрежови протоколи, сокети и комплект протоколи TCP/IP.	10
2.	Разбира HTTP протокол и прави разлика между методите GET и POST.	10
3.	Познава най-често използваните тагове и семантичните елементи в HTML. Познава техники за стилизиране на уеб страници чрез CSS.	10
4.	Разбира начин на работа на HTTP сървър.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 16: Интернет програмиране

План-тезис: Уеб MVC платформа, архитектура и компоненти. Свързване на HTTP сървър с backend език за програмиране. Комуникация с база от данни. Шаблонни езици от страна на сървъра. Създаване на MVC уеб приложения. Управление на състоянието в уеб приложенията. Автентикация и оторизация. Сигурност на уеб приложенията и добре известни атаки.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се коригират следните проблеми в предоставения проект:

1. Проектът не се стартира. Намерете решение за синтактичните и логически грешки, за да се стартира проекта успешно
2. При създаване на нов запис за книга важат валидации, но при редактиране на запис същите липсват. Решете проблема като добавите валидации и при редакция

3. При опит за редакция на съществуващ запис, вместо това се създава нов запис със редактираните данни
4. В изгледа, показващ всички записи на книги има грешка при показването на записите – вместо цена се показва заглавието на книгата:

Тестове:

Приложението работи и 4те описани дефекта са коригирани..

Фрагмент:

Book.cs

```
public class Book
{
    public int Id { get; set; }

    public string Author { get; set; }

    public string Title { get; set; }

    public decimal Price { get; set; }

    public string ISBN { get; set; }

    public int Pages { get; set; }
}
```

MyLibraryDb.cs

```
public class MyLibraryDb : DbContext
{
    public virtual DbSet<Book> Books { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Data Source=(localdb)\ProjectsV13;Initial
Catalog=MyLibraryDb;");
        optionsBuilder.UseLazyLoadingProxies();
    }
}
```

BooksController.cs

```
public class BooksController : Controller
{
    private readonly MyLibraryDb _context;
    private const int PageSize = 10;
```

```
    public BooksController()
    {
        _context = new MyLibraryDb();
    }

    // GET: Books
    public async Task<IActionResult> Index(BooksIndexViewModel model)
    {
        model.Pager ??= new PagerViewModel();
        model.Pager.CurrentPage = model.Pager.CurrentPage <= 0 ? 1 :
model.Pager.CurrentPage;

        List<BooksViewModel> items = await
_context.Books.Skip((model.Pager.CurrentPage - 1) * PageSize).Take(PageSize).Select(b
=> new BooksViewModel()
        {
            Id = b.Id,
            Author = b.Author,
            ISBN = b.ISBN,
            Pages = b.Pages,
            Price = b.Price,
            Title = b.Title

        }).ToListAsync();

        model.Items = items;
        model.Pager.PagesCount = (int)Math.Ceiling(await _context.Books.CountAsync()
/ (double)PageSize);

        return View(model);
    }

    // GET: Books/Create
    public IActionResult Create()
    {
        BooksCreateViewModel model = new BooksCreateViewModel();

        return View(model);
    }

    // POST: Books/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create(BooksCreateViewModel model)
    {
        if (ModelState.IsValid)
        {
            Book book = new Book
            {
                Price = model.Price,
                ISBN = model.ISBN,
                Author = model.Author,
                Pages = model.Pages,
                Title = model.Title
            };

            _context.Add(book);
            await _context.SaveChangesAsync();
```

```
            return RedirectToAction(nameof(Index));
        }

        return View(model);
    }

    // GET: Books/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        Book book = await _context.Books.FindAsync(id);
        if (book == null)
        {
            return NotFound();
        }

        BooksEditViewModel model = new BooksEditViewModel
        {
            Id = book.Id,
            ISBN = book.ISBN,
            Author = book.Author,
            Pages = book.Pages,
            Title = book.Title,
            Price = book.Price
        };

        return View(model);
    }

    // POST: Books/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(BooksEditViewModel model)
    {
        if (ModelState.IsValid)
        {
            Book book = new Book
            {
                Id = model.Id,
                ISBN = model.ISBN,
                Author = model.Author,
                Pages = model.Pages,
                Title = model.Title,
                Price = model.Price
            };

            try
            {
                _context.Update(book);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
```

```
                if (!BookExists(book.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return RedirectToAction(nameof(Index));
        }

        return View(model);
    }

    // GET: Books/Delete/5
    public async Task<IActionResult> Delete(int id)
    {
        Book book = await _context.Books.FindAsync(id);
        _context.Books.Remove(book);
        await _context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));
    }

    private bool BookExists(int id)
    {
        return _context.Books.Any(e => e.Id == id);
    }
}
```

HomeController.cs

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return RedirectToAction("Index", "Books");
    }
}
```

BooksCreateViewModel.cs

```
public class BooksCreateViewModel
{
    [Required]
    [MaxLength(80, ErrorMessage = "Author name cannot be longer than 80 characters")]
    public string Author { get; set; }

    [Required]
    [MaxLength(80, ErrorMessage = "Title length cannot be more than 80 characters")]
    public string Title { get; set; }
```

```
    [Required]
    [Range(0, double.MaxValue, ErrorMessage = "Negative values are not accepted")]
    public decimal Price { get; set; }

    [Required]
    [RegularExpression(@"^\d{10}$|^\d{13}$", ErrorMessage = "ISBN should be either 10
or 13 characters long")]
    public string ISBN { get; set; }

    [Required]
    [Range(1, int.MaxValue, ErrorMessage = "Negative values or 0 are not accepted")]
    public int Pages { get; set; }
}
```

BooksEditViewModel.cs

```
public class BooksEditViewModel
{
    [HiddenInput]
    public int Id { get; set; }

    [Required]
    [MaxLength(80, ErrorMessage = "Author name cannot be longer than 80 characters")]
    public string Author { get; set; }

    [Required]
    [MaxLength(80, ErrorMessage = "Title length cannot be more than 80 characters")]
    public string Title { get; set; }

    [Required]
    [Range(0, double.MaxValue, ErrorMessage = "Negative values are not accepted")]
    public decimal Price { get; set; }

    [Required]
    [RegularExpression(@"^\d{10}$|^\d{13}$", ErrorMessage = "ISBN should be either 10
or 13 characters long")]
    public string ISBN { get; set; }

    [Required]
    [Range(1, int.MaxValue, ErrorMessage = "Negative values or 0 are not accepted")]
    public int Pages { get; set; }
}
```


BooksIndexViewModel.cs

```
public class BooksIndexViewModel
{
    public PagerViewModel Pager { get; set; }

    public ICollection<BooksViewModel> Items { get; set; }
```

```
}
```

BooksViewModel.cs

```
public class BooksViewModel
{
    public int Id { get; set; }
    public string Author { get; set; }
    public string Title { get; set; }
    public decimal Price { get; set; }
    public string ISBN { get; set; }
    public int Pages { get; set; }
}
```

PagerViewModel.cs

```
public class PagerViewModel
{
    public int CurrentPage { get; set; }
    public int PagesCount { get; set; }
}
```


ErrorViewModel.cs

```
public class ErrorViewModel
{
    public string RequestId { get; set; }
    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
}
```

Create.cshtml

```
@model Web.Models.Books.BooksCreateViewModel

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>Book</h4>
<hr />
<div class="row">
```

```
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Author" class="control-label"></label>
                <input asp-for="Author" class="form-control" />
                <span asp-validation-for="Author" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ISBN" class="control-label"></label>
                <input asp-for="ISBN" class="form-control" />
                <span asp-validation-for="ISBN" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Pages" class="control-label"></label>
                <input asp-for="Pages" class="form-control" />
                <span asp-validation-for="Pages" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Edit.cshtml

```
@model Web.Models.Books.BooksEditViewModel

@{
    ViewData["Title"] = "Edit";
}
<h1>Edit</h1>
<h4>Book</h4>
<hr />
<div class="row">
    <div class="col-md-4">
```

```
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="Author" class="control-label"></label>
                <input asp-for="Author" class="form-control" />
                <span asp-validation-for="Author" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ISBN" class="control-label"></label>
                <input asp-for="ISBN" class="form-control" />
                <span asp-validation-for="ISBN" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Pages" class="control-label"></label>
                <input asp-for="Pages" class="form-control" />
                <span asp-validation-for="Pages" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```


Index.cshtml

```
@model Web.Models.Books.BooksIndexViewModel
@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
```

```
                Author
            </th>
            <th>
                Title
            </th>
            <th>
                Price
            <th>
                ISBN
            </th>
            <th>
                Page count
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Items)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Author)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Title)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Price)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ISBN)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Pages)
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.Id">Edit</a>
                    <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
                </td>
            </tr>
        }
    </tbody>
    <ul class="pagination">
        @for (var i = 1; i <= Model.Pager.PagesCount; i++)
        {
            <li class="page-item @(i == Model.Pager.CurrentPage ? "active" : "")">
                <a asp-route-Pager.CurrentPage="@i" class="page-link">@i</a>
            </li>
        }
    </ul>
</table>
```

_Layout.cshtml

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Web</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Books</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2019 - Web
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

_ValidationScriptsPartial.cshtml

```
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

Error.cshtml

```
@model ErrorViewModel
```

```
@{
    ViewData["Title"] = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed
information about the error that occurred.
</p>
<p>
    <strong>The Development environment shouldn't be enabled for deployed
applications.</strong>
    It can result in displaying sensitive information from exceptions to end users.
    For local debugging, enable the <strong>Development</strong> environment by
setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
<strong>Development</strong>
    and restarting the app.
</p>
```

_ViewImports.cshtml

```
@using Web
@using Web.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

_ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Program.cs

```
public class Program
{
```

```
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

Startup.cs

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the
container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllersWithViews();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();
```

```
        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
}
```

site.css

```
a.navbar-brand {
  white-space: normal;
  text-align: center;
  word-break: break-all;
}

/* Provide sufficient contrast against white background */
a {
  color: #0366d6;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

/* Sticky footer styles
-------------------------------------------------- */
html {
  font-size: 14px;
}
@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.border-top {
  border-top: 1px solid #e5e5e5;
}
.border-bottom {
  border-bottom: 1px solid #e5e5e5;
}

.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}
```

```
button.accept-policy {
  font-size: 1rem;
  line-height: inherit;
}

/* Sticky footer styles
-------------------------------------------------- */
html {
  position: relative;
  min-height: 100%;
}

body {
  /* Margin bottom by footer height */
  margin-bottom: 60px;
}
.footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px; /* Vertically center the text there */
}
```


№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава уеб MVC платформа, нейните архитектура и компоненти.	10
2.	Разбира връзката между HTTP сървър и сървърен (backend) език за програмиране, както и шаблонни езици от страна на сървъра.	10
3.	Познава техники за комуникация с база от данни.	10
4.	Разбира структурата на MVC уеб приложения, техники за управление на състоянието в тях. Различава автентикация и авторизация. Познава добрите практики за сигурност на уеб приложенията.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 17: Интернет програмиране

План-тезис: Създаване на REST API. Извикване на REST заявки с JavaScript и AJAX. Внедряване на проект (deployment).

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се идентифицират проблеми в приложния програмен интерфейс и да се отстранят.

Тестове:

Приложението работи коректно.

Фрагмент:

User.cs

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

ChatMessage.cs

```
public class ChatMessage
{
    public int Id { get; set; }
    public int SenderId { get; set; }
    public int ReceiverId { get; set; }
    public string Message { get; set; }
}
```

ChatAPIContext.cs

```
public class ChatAPIContext : DbContext
{
```

```
    public ChatAPIContext (DbContextOptions<ChatAPIContext> options)
        : base(options)
    {

    }

    public DbSet<ChatAPI.User> User { get; set; }

    public DbSet<ChatAPI.ChatMessage> ChatMessage { get; set; }
}
```

UsersController.cs

```
[Route("api/[controller]")]
[ApiController]
public class UsersController : ControllerBase
{
    private readonly ChatAPIContext _context;

    public UsersController(ChatAPIContext context)
    {
        _context = context;
    }

    // GET: api/Users
    [HttpGet]
    public async Task<ActionResult<IEnumerable<User>>> GetUser()
    {
        return await _context.User.ToListAsync();
    }

    // GET: api/Users/5
    [HttpGet("{id}")]
    public async Task<ActionResult<User>> GetUser(int id)
    {
        var user = await _context.User.FindAsync(id);

        if (user == null)
        {
            return NotFound();
        }

        return user;
    }

    // PUT: api/Users/5
    // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
    // more details see https://aka.ms/RazorPagesCRUD.
    [HttpPut("{id}")]
    public async Task<IActionResult> PutUser(int id, User user)
    {
        if (id != user.Id)
        {
```

```
            return BadRequest();
        }

        _context.Entry(user).State = EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!UserExists(id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return NoContent();
    }

    // POST: api/Users
    // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
    // more details see https://aka.ms/RazorPagesCRUD.
    [HttpPost]
    public async Task<ActionResult<User>> PostUser(User user)
    {
        _context.User.Add(user);
        await _context.SaveChangesAsync();

        return CreatedAtAction("GetUser", new { id = user.Id }, user);
    }

    // DELETE: api/Users/5
    [HttpDelete("{id}")]
    public async Task<ActionResult<User>> DeleteUser(int id)
    {
        var user = await _context.User.FindAsync(id);
        if (user == null)
        {
            return NotFound();
        }

        _context.User.Remove(user);
        await _context.SaveChangesAsync();

        return user;
    }

    private bool UserExists(int id)
    {
        return _context.User.Any(e => e.Id == id);
    }
}
```

ChatMessagesController.cs

```
[Route("api/[controller]")]
[ApiController]
public class ChatMessagesController : ControllerBase
{
    private readonly ChatAPIContext _context;

    public ChatMessagesController(ChatAPIContext context)
    {
        _context = context;
    }

    // GET: api/ChatMessages
    [HttpGet]
    public async Task<ActionResult<IEnumerable<ChatMessage>>> GetChatMessage()
    {
        return await _context.ChatMessage.ToListAsync();
    }

    // GET: api/ChatMessages/5
    [HttpGet("{id}")]
    public async Task<ActionResult<ChatMessage>> GetChatMessage(int id)
    {
        var chatMessage = await _context.ChatMessage.FindAsync(id);

        if (chatMessage == null)
        {
            return NotFound();
        }

        return chatMessage;
    }

    // PUT: api/ChatMessages/5
    // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
    // more details see https://aka.ms/RazorPagesCRUD.
    [HttpPut("{id}")]
    public async Task<IActionResult> PutChatMessage(int id, ChatMessage chatMessage)
    {
        if (id != chatMessage.Id)
        {
            return BadRequest();
        }

        _context.Entry(chatMessage).State = EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ChatMessageExists(id))
            {
```

```
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return NoContent();
    }

    // POST: api/ChatMessages
    // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
    // more details see https://aka.ms/RazorPagesCRUD.
    [HttpPost]
    public async Task<ActionResult<ChatMessage>> PostChatMessage(ChatMessage
chatMessage)
    {
        _context.ChatMessage.Add(chatMessage);
        await _context.SaveChangesAsync();

        return CreatedAtAction("GetChatMessage", new { id = chatMessage.Id },
chatMessage);
    }

    // DELETE: api/ChatMessages/5
    [HttpDelete("{id}")]
    public async Task<ActionResult<ChatMessage>> DeleteChatMessage(int id)
    {
        var chatMessage = await _context.ChatMessage.FindAsync(id);
        if (chatMessage == null)
        {
            return NotFound();
        }

        _context.ChatMessage.Remove(chatMessage);
        await _context.SaveChangesAsync();

        return chatMessage;
    }

    private bool ChatMessageExists(int id)
    {
        return _context.ChatMessage.Any(e => e.Id == id);
    }
}
```


user.js

```
class User {
    constructor(username, password, firstName, lastName, isAdmin) {
        this.username = username;
        this.password = password;
        this.firstName = firstName;
```

```
        this.lastName = lastName;
        this.isAdmin = isAdmin;
    }

    get _id() {
        return this.id;
    }

    set _id(x) {
        this.id = x;
    }

    get _username() {
        return this.username;
    }

    set _username(x) {
        this.username = x;
    }

    get _password() {
        return this.password;
    }

    set _password(x) {
        this.password = x;
    }

    get _firstName() {
        return this.firstName;
    }

    set _firstName(x) {
        this.firstName = x;
    }

    get _lastName() {
        return this.lastName;
    }

    set _lastName(x) {
        this.lastName = x;
    }
}
```


chatMessage.js

```
class ChatMessage {
    constructor(senderId, receiverId, message) {
        this.senderId = senderId;
        this.receiverId = receiverId;
        this.message = message;
    }

    get _id() {
```

```
        return this.id;
    }

    set _id(x) {
        this.id = x;
    }

    get _senderId() {
        return this.senderId;
    }

    set _senderId(x) {
        this.senderId = x;
    }

    get _receiverId() {
        return this.receiverId;
    }

    set _receiverId(x) {
        this.receiverId = x;
    }

    get _message() {
        return this.message;
    }

    set _message(x) {
        this.message = x;
    }
}
```


userRepo.js

```
class UsersRepository {

    static async getAll() {

        const response = await fetch(BaseUrl + UsersUrl, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
        });

        const result = await response.json();

        return result;
    }

    static async getById(id) {

        const response = await fetch(BaseUrl + UsersUrl + "/" + id, {
```

```
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
        })

        const result = await response.json();

        return result;
    }

    static async addUser(item) {

        await fetch(BaseUrl + UsersUrl, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
            body: JSON.stringify(item)
        });
    }

    static async editUser(item) {

        await fetch(BaseUrl + UsersUrl, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
            body: JSON.stringify(item)
        });
    }

    static async deleteUser(id) {

        await fetch(BaseUrl + UsersUrl + '/' + id, {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
        });
    }
}
```


chatMessagesRepo.js

```
class ChatMessageRepository {

    static async getAll() {

        const response = await fetch(BaseUrl + ChatMessageUrl, {
```

```
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
        });

        const result = await response.json();

        return result;
    }

    static async getById(id) {

        const response = await fetch(BaseUrl + ChatMessageUrl + "/" + id, {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
        })

        const result = await response.json();

        return result;
    }

    static async addChatMessage(item) {

        await fetch(BaseUrl + ChatMessageUrl, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
            body: JSON.stringify(item)
        });
    }

    static async editChatMessage(item) {

        await fetch(BaseUrl + ChatMessageUrl, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
            },
            body: JSON.stringify(item)
        });
    }

    static async deleteChatMessage(id) {

        await fetch(BaseUrl + ChatMessageUrl + '/' + id, {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': await AuthenticationService.getAuthorizationHeader()
```

```
            },
        });
    }
}
```

usersPage.js

```
function usersPage() {
    return `<button id="newUserLink" onclick="usersEditLink_Click()">New</button>
    <table id="usersTable">
        <tr>
            <td>Username</td>
            <td>Password</td>
            <td>First Name</td>
            <td>Last Name</td>
            <td>Is admin</td>
            <td></td>
            <td></td>
        </tr>
    </table>`;
}
```

usersEditPage.js

```
function usersEditPage() {
    return `<input type="hidden" id="id" name="id" />
    <fieldset>
        <legend>User</legend>
        <table>
            <tr>
                <td>Username:</td>
                <td><input type="text" id="username" name="username" /></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="password" id="password" name="password" /></td>
            </tr>
            <tr>
                <td>First Name:</td>
                <td><input type="text" id="firstName" name="firstName" /></td>
            </tr>
            <tr>
                <td>Last Name:</td>
                <td><input type="text" id="lastName" name="lastName" /></td>
            </tr>
            <tr>
                <td colspan="2"><input type="button" onclick="usersEditForm_Submit()"
value="Save" /></td>
            </tr>
        </table>
    </fieldset>`;
}
```

chatMessagesPage.js

```
function chatMessagePage() {
    return `<button id="newMessageLink"
onclick="messagesEditLink_Click()">New</button>
    <table id="contactsTable">
        <tr>
            <td>From</td>
            <td>To</td>
            <td>Message</td>
            <td></td>
            <td></td>
            <td></td>
        </tr>
    </table>`;
}
```


chatMessagesEditPage.js

```
function chatMessagesEditPage() {
    return `<input type="hidden" id="id" name="id" />
    <input type="hidden" id="senderId" name="senderId" />
    <fieldset>
        <legend>Message</legend>
        <table>
            <tr>
                <td>Message:</td>
                <td><input type="text" id="message" name="message" /></td>
            </tr>
            <tr>
                <td colspan="2"><input type="button"
onclick="chatMessageEditForm_Submit()" value="Save" /></td>
            </tr>
        </table>
    </fieldset>`;
}
```

events.js

```
async function usersLink_Click() {

    await render(usersPage());
    const usersTable = document.getElementById('usersTable');

    const items = await UsersRepository.getAll();
    if (items == null)
        return;

    for (let i = 0; i < items.length; i++) {
        const currentItem = items[i];
```

```
        const tr = document.createElement('TR');

        const usernameTd = document.createElement('TD');
        usernameTd.innerHTML = currentItem.username;

        const passwordTd = document.createElement('TD');
        passwordTd.innerHTML = currentItem.password;

        const firstNameTd = document.createElement('TD');
        firstNameTd.innerHTML = currentItem.firstName;

        const lastNameTd = document.createElement('TD');
        lastNameTd.innerHTML = currentItem.lastName;

        const isAdminTd = document.createElement('TD');
        isAdminTd.innerHTML = currentItem.isAdmin;

        const editTd = document.createElement('TD');
        const editButton = document.createElement('BUTTON');
        editButton.innerHTML = 'EDIT';
        editButton.addEventListener('click', () =>
usersEditButton_Click(currentItem.id));
        editTd.appendChild(editButton);

        const deleteTd = document.createElement('TD');
        const deleteButton = document.createElement('BUTTON');
        deleteButton.innerHTML = 'DELETE';
        deleteButton.addEventListener('click', () =>
usersDeleteButton_Click(currentItem.id));
        deleteTd.appendChild(deleteButton);

        tr.appendChild(usernameTd);
        tr.appendChild(passwordTd);
        tr.appendChild(firstNameTd);
        tr.appendChild(lastNameTd);
        tr.appendChild(isAdminTd);
        tr.appendChild(editTd);
        tr.appendChild(deleteTd);

        usersTable.appendChild(tr);
    }
}

async function usersEditLink_Click() {
    await render(usersEditPage());
}

async function usersEditButton_Click(id) {

    await usersEditLink_Click();
    const item = await UsersRepository.getById(id);

    document.getElementById('id').value = item.id;
    document.getElementById('username').value = item.username;
    document.getElementById('password').value = item.password;
    document.getElementById('firstName').value = item.firstName;
    document.getElementById('lastName').value = item.lastName;
```

```
}

async function usersEditForm_Submit() {
    const id = document.getElementById('id').value;
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
    const firstName = document.getElementById('firstName').value;
    const lastName = document.getElementById('lastName').value;

    const item = new User(username, password, firstName, lastName, false);

    if (id == "") {
        await UsersRepository.addUser(item);
    } else {
        item.id = id;
        await UsersRepository.editUser(item);
    }

    await usersLink_Click();
}

async function usersDeleteButton_Click(id) {

    await UsersRepository.deleteUser(id);
    await usersLink_Click();
}
...
```

index.js

```
(async() => {

    await render(homePage());
    await handleMenu();
})();

async function render(innerHtml) {
    let contentDiv = await document.getElementById('content');
    contentDiv.innerHTML = innerHtml;
}

async function handleMenu() {

    const loggedUser = await AuthenticationService.getLoggedUser();

    if (loggedUser == null) {
        document.getElementById('loginLink').style.display = '';
        document.getElementById('homeLink').style.display = '';
        document.getElementById('usersLink').style.display = 'none';
        document.getElementById('chatMessagesLink').style.display = 'none';
        document.getElementById('logoutLink').style.display = 'none';
        return;
    } else {
        document.getElementById('logoutLink').style.display = '';
        document.getElementById('loginLink').style.display = 'none';
```

```
    }

    if (loggedUser.isAdmin) {

        document.getElementById('usersLink').style.display = '';
        document.getElementById('chatMessagesLink').style.display = '';
    } else {
        document.getElementById('chatMessagesLink').style.display = '';
        document.getElementById('usersLink').style.display = 'none';
    }
}
```

services.js

```
class AuthenticationService {

    static async authenticate(username, password) {

        const response = await fetch(BaseUrl + AuthUrl, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                username: username,
                password: password
            })
        });

        if (response.ok) {
            const result = await response.json();
            const token = result.token;
            const loggedUserId = result.userId;

            await window.sessionStorage.setItem(Token, token);

            const loggedUser = await UsersRepository.getById(loggedUserId);
            await window.sessionStorage.setItem(LoggedUser, JSON.stringify({
                id: loggedUserId,
                isAdmin: loggedUser.isAdmin
            }))
        }
    }

    static async getLoggedUser() {

        return JSON.parse(await window.sessionStorage.getItem(LoggedUser));
    }

    static async logout() {

        await window.sessionStorage.removeItem(LoggedUser);
        await window.sessionStorage.removeItem(Token);
    }
```

```
    static async getAuthorizationHeader() {

        const token = await window.sessionStorage.getItem(Token);
        return 'Bearer ' + await window.sessionStorage.getItem(Token);
    }
}
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава техники за създаване на REST API.	10
2.	Разбира извикване на REST заявки с JavaScript и AJAX.	10
3.	Познава процеса на документиране на REST API	10
4.	Описва внедряване на проект (deployment).	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

Изпитна тема № 18: Софтуерно инженерство

План-тезис: Етапи в софтуерната разработка. Методологии за разработка на софтуер. Инструменти. Сорс-контрол системи. Софтуерни изисквания и прототипи. Софтуерно тестване. Софтуерна документация.

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Да се коригират следните проблеми в предоставения проект:

1. При създаване на нов запис възниква грешка със следното съобщение: „InvalidOperationException: The entity type 'ContactsCreateViewModel' was not found.

Ensure that the entity type has been added to the model.“ Съответно записът не се запазва в базата от данни

2. Данните на вече запезените конакти не се показват коректно – на мястото на бележката се показва едно и също съобщение за всички контакти, а на мястото на името им – бележката:
3. При създаване, редакция на контакт полето за бележка – Note е задължително, а не трябва да бъде такова
4. При редакция на контакт възниква грешка със следното съобщение „SqlException: Cannot insert explicit value for identity column in table 'Contacts' when IDENTITY_INSERT is set to OFF.“

Тестове:

Приложението работи и 4те описани дефекта са коригирани..

Фрагмент:

Contact.cs

```
public class Contact
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string PhoneNumber { get; set; }
    public string Email { get; set; }
    public string Note { get; set; }
}
```


site.css

```
a.navbar-brand {
  white-space: normal;
  text-align: center;
  word-break: break-all;
}

/* Provide sufficient contrast against white background */
a {
  color: #0366d6;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
```

```
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

/* Sticky footer styles
-------------------------------------------------- */
html {
  font-size: 14px;
}
@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.border-top {
  border-top: 1px solid #e5e5e5;
}
.border-bottom {
  border-bottom: 1px solid #e5e5e5;
}

.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

button.accept-policy {
  font-size: 1rem;
  line-height: inherit;
}

/* Sticky footer styles
-------------------------------------------------- */
html {
  position: relative;
  min-height: 100%;
}

body {
  /* Margin bottom by footer height */
  margin-bottom: 60px;
}
.footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px; /* Vertically center the text there */
}
```


ContactsController.cs

```
public class ContactsController : Controller
{
```

```
    private const int PageSize = 10;
    private readonly ContactsDb _context;

    public ContactsController()
    {
        _context = new ContactsDb();
    }

    // GET: Contacts
    public async Task<IActionResult> Index(ContactsIndexViewModel model)
    {
        model.Pager ??= new PagerViewModel();
        model.Pager.CurrentPage = model.Pager.CurrentPage <= 0 ? 1 :
model.Pager.CurrentPage;

        List<ContactsViewModel> items = await
_context.Contacts.Skip((model.Pager.CurrentPage - 1) *
PageSize).Take(PageSize).Select(c => new ContactsViewModel()
        {
            Id = c.Id,
            Email = c.Email,
            Name = c.Name,
            Note = c.Note,
            PhoneNumber = c.PhoneNumber

        }).ToListAsync();

        model.Items = items;
        model.Pager.PagesCount = (int)Math.Ceiling(await
_context.Contacts.CountAsync() / (double)PageSize);

        return View(model);
    }

    // GET: Contacts/Create
    public IActionResult Create()
    {
        ContactsCreateViewModel model = new ContactsCreateViewModel();

        return View(model);
    }

    // POST: Contacts/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create(ContactsCreateViewModel model)
    {
        if (ModelState.IsValid)
        {
            Contact contact = new Contact
            {
                Note = model.Note,
                Email = model.Email,
                Name = model.Name,
                PhoneNumber = model.PhoneNumber
            };

            _context.Add(contact);
```

```
            await _context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));
        }

        return View(model);
    }

    // GET: Contacts/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        Contact contact = await _context.Contacts.FindAsync(id);
        if (contact == null)
        {
            return NotFound();
        }

        ContactsEditViewModel model = new ContactsEditViewModel
        {
            Id = contact.Id,
            Email = contact.Email,
            Name = contact.Name,
            PhoneNumber = contact.PhoneNumber,
            Note = contact.Note

        };

        return View(model);
    }

    // POST: Contacts/Edit/5
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(ContactsEditViewModel model)
    {
        if (ModelState.IsValid)
        {
            Contact contact = new Contact
            {
                Id = model.Id,
                Email = model.Email,
                Name = model.Name,
                PhoneNumber = model.PhoneNumber,
                Note = model.Note
            };

            try
            {
                _context.Update(contact);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
```

```
                if (!ContactExists(contact.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return RedirectToAction(nameof(Index));
        }

        return View(model);
    }

    // GET: Contacts/Delete/5
    public async Task<IActionResult> Delete(int id)
    {
        Contact contact = await _context.Contacts.FindAsync(id);
        _context.Contacts.Remove(contact);
        await _context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));
    }

    private bool ContactExists(int id)
    {
        return _context.Contacts.Any(e => e.Id == id);
    }
}
```

HomeController.cs

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return RedirectToAction("Index", "Contacts");
    }
}
```

ContactsCreateViewModel.cs

```
public class ContactsCreateViewModel
{
    [Required]
    [MaxLength(50, ErrorMessage = "Name can be at most 50 characters long")]
    public string Name { get; set; }

    [Required]
    [RegularExpression(@"^\d{10}$", ErrorMessage = "Invalid phone number")]
```

```
    public string PhoneNumber { get; set; }

    [Required]
    [RegularExpression(@"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-
]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$", ErrorMessage = "Invalid email")]
    public string Email { get; set; }

    public string Note { get; set; }
}
```

ContactsEditViewModel.cs

```
public class ContactsEditViewModel
{
    public int Id { get; set; }

    [Required]
    [MaxLength(50, ErrorMessage = "Name can be at most 50 characters long")]
    public string Name { get; set; }

    [Required]
    [RegularExpression(@"^\d{10}$", ErrorMessage = "Invalid phone number")]
    public string PhoneNumber { get; set; }

    [Required]
    [RegularExpression(@"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-
]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$", ErrorMessage = "Invalid email")]
    public string Email { get; set; }

    public string Note { get; set; }
}
```


ContactsIndexViewModel.cs

```
public class ContactsIndexViewModel
{
    public PagerViewModel Pager { get; set; }
    public ICollection<ContactsViewModel> Items { get; set; }
}
```

ContactsViewModel.cs

```
public class ContactsViewModel
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string PhoneNumber { get; set; }
    public string Email { get; set; }
    public string Note { get; set; }
}
```

PagerViewModel.cs

```
public class PagerViewModel
{
    public int CurrentPage { get; set; }
    public int PagesCount { get; set; }
}
```

ErrorViewModel.cs

```
public class ErrorViewModel
{
    public string RequestId { get; set; }
    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
}
```

Index.cshtml

```
@model Web.Models.Contacts.ContactsIndexViewModel
@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                Name
            </th>
            <th>
                Phone number
            </th>
            <th>
                Email
            </th>
            <th>
                Note
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Items)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
```

```
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.PhoneNumber)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Email)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Note)
                </td>
                <td>
                    <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                    <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
                </td>
            </tr>
        }
    </tbody>
    <ul class="pagination">
        @for (var i = 1; i <= Model.Pager.PagesCount; i++)
        {
            <li class="page-item @(i == Model.Pager.CurrentPage ? "active" : "")">
                <a asp-route-Pager.CurrentPage="@i" class="page-link">@i</a>
            </li>
        }
    </ul>
</table>
```

Edit.cshtml

```
@model Web.Models.Contacts.ContactsEditViewModel
@{
    ViewData["Title"] = "Edit";
}
<h1>Edit</h1>

<h4>Contact</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PhoneNumber" class="control-label"></label>
                <input asp-for="PhoneNumber" class="form-control" />
                <span asp-validation-for="PhoneNumber" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
```

```
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Note" class="control-label"></label>
                <input asp-for="Note" class="form-control" />
                <span asp-validation-for="Note" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Create.cshtml

```
@model Web.Models.Contacts.ContactsCreateViewModel
@{
    ViewData["Title"] = "Create";
}
<h1>Create</h1>
<h4>Contact</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PhoneNumber" class="control-label"></label>
                <input asp-for="PhoneNumber" class="form-control" />
                <span asp-validation-for="PhoneNumber" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Email" class="control-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Note" class="control-label"></label>
                <input asp-for="Note" class="form-control" />
                <span asp-validation-for="Note" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
```

```
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Web</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
white border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Contacts</a>
                <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2019 - Web
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

_ValidationScriptsPartial.cshtml

```
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

Error.cshtml

```
@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}
<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}
<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed
information about the error that occurred.
</p>
<p>
    <strong>The Development environment shouldn't be enabled for deployed
applications.</strong>
    It can result in displaying sensitive information from exceptions to end users.
    For local debugging, enable the <strong>Development</strong> environment by
setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
<strong>Development</strong>
    and restarting the app.
</p>
```


_ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Program.cs

```
public class Program
{
    public static void Main(string[] args)
    {
```

```
        CreateHostBuilder(args).Build().Run();
    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

Startup.cs

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the
container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllersWithViews();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
```

```
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
}
```

№	*Критерии за оценяване*	*Максимален брой точки*
1.	Познава етапите в софтуерната разработка.	10
2.	Разбира методологии за разработка на софтуер, инструменти и сорс-контрол системи.	10
3.	Описва софтуерни изисквания и прототипи.	10
4.	Разбира принципите на софтуерното тестване и документиране.	10
5.	Решава приложната задача/казус.	20
	Общ брой точки:	**60**

IV. ДЪРЖАВЕН ИЗПИТ ПО ПРАКТИКА НА ПРОФЕСИЯТА И СПЕЦИАЛНОСТТА

1. Указания за съдържанието на индивидуалните практически задания

Чрез държавния изпит по практика на професията и специалността се проверяват и оценяват професионалните умения и компетентности на обучаваните, отговарящи на трета степен на професионална квалификация.

Изпитът по практика на професията и специалността се състои в създаване на цялостна уеб платформа, която задължително включва следните три компонента:

1. визуален интерфейс
2. система за управление на бази от данни
3. сървърна система, базирана на трислойния модел, която координира визуалния интерфейс и системата за управление на бази от данни

Индивидуалните практически задания се разработват от комисия, назначена със заповед на директора/ръководителя. Индивидуалното изпитно задание съдържа пълното наименование на училището/обучаващата институция, празни редове за попълване имената на обучавания, квалификационната форма, началната дата и началния час на изпита, краен срок на изпита – дата и час, темата на индивидуалното практическо задание и изискванията към крайния резултат от изпълнението на заданието. По решение на комисията могат да се дадат допълнителни указания, които да подпомогнат обучавания при изпълнение на индивидуалното практическо задание.

Индивидуалните практически задания се изготвят от комисията за провеждане и оценяване на изпита по практика на професията и специалността в училището/обучаващата институция. Броят на изготвените задания трябва да бъде поне с един повече от броя на

явяващите се в деня на изпита. Всеки обучаван изтегля индивидуалното си практическо задание, в което веднага саморъчно написва трите си имена.

2. Критерии за оценяване

За всяко индивидуално практическо задание комисията за провеждане и оценяване на изпита по практика на професията и специалността разработва показатели по критериите, определени в таблицата.

№	КРИТЕРИИ	ПОКАЗАТЕЛИ	Максимален брой точки
1.	Умения за работа с HTML и CSS	● оформление на страница - позициониране на елементи, чрез HTML тагове (например: div) и CSS атрибути (например: float, clear и display) ● използване на семантични HTML тагове ● използване на CSS селектори, изнасяне на CSS във външен файл, използване на класове в HTML кода	**5** **5** **5**
2.	Умения за работа с бази от данни	● моделиране на база от данни, чрез директно използване на SQL диалект или библиотека (ORM) за генериране на заявки с програмен код. Структурата да съдържа поне две таблици с поне една връзка от тип едно към много и поне една от таблиците да има колона от тип дата	**10**

		● Добавяне на данни в базата от данни, чрез директно използване на SQL диалект или сървърната система (компонент 3 от изпита по практика)	**10**
3.	Умения за разработка на сървърни системи, базирани на трислойния модел	● Да има потребителски екрани за извършване на основните операции с данните от базата (CRUD - създаване, визуализиране, редакция и изтриване) ● Да има възможност за качване на файл с изображение с логическа връзка към данните от базата Пример: Аватар на потребител, снимка на продукт и т.н. ● да има заглавна страница със статистическа информация, базирана на агрегирани данни от базата	**10** **5** **5**

4.	**Здравословно и безопасно упражняване на професията.**	● знания и умения за безопасна работа на работното място ● умение за адекватно реагиране в критични ситуации в рамките на компетенциите си ● умение за оказване на първа помощ на пострадал при авария (при токов удар, пожар, наранявания и др.)	**3**
5.	**Професионално-личностни качества.**	● отговорност към извършената работа ● трудова и технологична дисциплина - създаване на четим и ясен код, а в по-комплексни ситуации, детайлно описание, посредством коментари	**2**
		Общ брой точки	**60**

Посочва се максималният брой точки, които се поставят при пълно, вярно и точно изпълнение на показателя. Те са в съответствие с посочените в Държавното образователно изискване за придобиване квалификация по професия код **481030** „Приложен програмист“, специалност код **4810301** „Приложно програмиране“.

V. СИСТЕМА ЗА ОЦЕНЯВАНЕ

Максималният брой точки за всяка изпитна тема или за всяко изпитно задание е 60. Неправилният отговор се оценява с 0 точки. Непълният отговор се оценява с част от точките за верен и пълен отговор.

Преминаването от точки в цифрова оценка съгласно чл. 7, ал. 4 от Наредба № 3 от 2003 г. за системата на оценяване:

Цифрова оценка = общият брой точки от всички критерии: 10

Получената цифрова оценка се изчислява с точност до 0,01.

Оценяването на писмените работи от държавния изпит по теория е в съответствие с чл. 46 от Наредба № 3 от 2003 г. за системата на оценяване.

Изпълнението на практическото задание от държавния изпит по практика се оценява в съответствие с чл. 48 от Наредба № 3 от 2003 г. за системата на оценяване.

VI. ПРЕПОРЪЧИТЕЛНА ЛИТЕРАТУРА

1. Наков, С., В. Колев и колектив. Въведение в със C#, София, 2015, ISBN 978-954-400-527-6, http://www.introprogramming.info/intro-csharp-book/
2. Наков, С. и колектив, Въведение в с Java, София, 2008, ISBN 978-954-400-055-4, http://www.introprogramming.info/intro-java-book/
3. Abelson, H., G. Sussman. Structure and Interpretation of Computer Programs, MIT Press, London, 1996
4. McLaughlin, B. D.; Pollice, G. & West, D. (2007), Head first object-oriented analysis and design - a brain-friendly guide to OOA&D., O'Reilly
5. Gamma, e. a. (1994), Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley Professional
6. Weisfeld, M. (2013), The Object-Oriented Thought Process (Developer's Library) 4th Edition, Addison-Wesley Professional
7. Beaulieu, A., Learning SQL: Master SQL Fundamentals, O'Reilly Media; 2nd edition (2009)
8. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring 2nd edition, John F. Dooley, Apress, 2017, ISBN 978-1484231524
9. Extreme Programming Explained: Embrace Change, 2nd Edition, Kent Beck, Addison-Wesley, 2005, ISBN 978-0321278654
10. Modern Operating Systems (4th Edition), Andrew S. Tanenbaum, Pearson, 2014, ISBN 978-0133591620
11. Operating Systems Concepts, Abraham Silberschatz, Greg Gagne, Peter Baer Galvin, Wiley, 2012, ISBN 978-1118063330
12. Operating Systems: Design and Implementation, 3rd edition, Albert S. Woodhull, Andrew S. Tanenbaum, Pearson, 2006, ISBN 978-0136373315
13. Exploring Arduino: Tools and Techniques for Engineering Wizardry, Jeremy Blum, Wiley, 2013, ISBN 978-1118549360
14. Programming Arduino: Getting Started with Sketches, Second Edition, Simon Monk, McGraw-Hill Education, 2016, ISBN 978-1259641633
15. Make: Arduino Bots and Gadgets: Six Embedded Projects with Open Source Hardware and Software, Tero Karvinen, Kimmo Karvinen, Maker Media, 2011, ISBN 978-1449389710
16. Hal Abelson's, Jerry Sussman's and Julie Sussman's Structure and Interpretation of Computer Programs, MIT Press, 1984; ISBN 0-262-01077-1, https://mitpress.mit.edu/sicp/
17. Miran Lipovača, Learn You a Haskell, No Starch Press, ISBN-13: 978-1-59327-283-8, http://learnyouahaskell.com
18. Paul Chiusano, Rúnar Bjarnason, Functional Programming in Principles with Scala, Manning Publications, ISBN-13: 978-1617290657, https://www.manning.com/books/functional-programming-in-scala
19. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring 2nd edition, John F. Dooley, Apress, 2017, ISBN 978-1484231524
20. Refactoring: Improving the Design of Existing Code (2nd Edition), Martin Fowler, Addison-Wesley, 2018, ISBN 978-0134757599
21. The Pragmatic Programmer, Andy Hunt, Dave Thomas, Addison-Wesley, 1999, ISBN 978-0201616224
22. Clean Code, Robert Martin, Prentice Hall, 2008, ISBN 978-0132350884
23. Code Complete: A Practical Handbook of Software Construction, Second Edition, Steve McConnell, Microsoft Press, 2004, 978-0735619678

Електронни информационни източници:

1. Портал за еОбучение по специалност „Приложен програмист“ https://it-kariera.mon.bg/e-learning
2. Програмиране 101 към ХакБългария https://github.com/HackBulgaria/Programming101-Python-2016

VII. АВТОРСКИ КОЛЕКТИВ

1. доц. д-р Димитър Минчев – Бургаски Свободен Университет
2. доц. д-р Ивайло Старибратов – ПУ „Паисий Хилендарски“
3. гл. ас. д-р Никола Вълчанов – ПУ „Паисий Хилендарски“
4. Ангел Георгиев – МусалаСофт АД, София
5. Веселина Карапеева – МГ „Акад. Кирил Попов“, Пловдив
6. Петър Петров – ПГЕЕ „Константин Фотинов“, Бургас
7. Росен Вълчев – МГ „Акад. Кирил Попов“, Пловдив
8. Хриси Плачкова – МГ „Акад. Кирил Попов“, Пловдив

VIII. ПРИЛОЖЕНИЯ

а) Примерен изпитен билет

.....

(пълно наименование на училището/обучаващата институция)

ДЪРЖАВЕН ИЗПИТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА И СПЕЦИАЛНОСТТА

ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН

НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ

по професия код 481030 „Приложен програмист“

специалност код 4810301 „Приложно програмиране“

Изпитен билет №……...

Изпитна тема: ……

(изписва се точното наименование на темата)

План-тезис: *……*

……

……

Приложна задача:

…… …… …

Описание на дидактическите материали:……..

Председател на изпитната комисия*:..... .....*

(име, фамилия) (подпис)

Директор/ръководител на обучаващата институция*:..... ……..*

(име, фамилия) (подпис)

(печат на училището/обучаващата институция)

б) Примерно индивидуално практическо задание

…….....

(пълно наименование на училището/обучаващата институция)

ДЪРЖАВЕН ИЗПИТ ПО ПРАКТИКА НА ПРОФЕСИЯТА И СПЕЦИАЛНОСТТА ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ

по професия код 481030 „Приложен програмист“

специалност код 4810301 „Приложно програмиране“

И н д и в и д у а л н о п р а к т и ч е с к о з а д а н и е № . . . . . .

На ученика/обучавания .....

(трите имена на ученика/обучавания)

от .....клас/курс,

начална дата на изпита: …… начален час: .....……

крайна дата на изпита: ..... час на приключване на изпита:.....

1. Да се .....: .....

(вписва се темата на изпитното задание)

……

1. *Указания (инструкции/изисквания) за изпълнение на практическото задание:*

……

……

……

УЧЕНИК/ОБУЧАВАН: …… .....

(име, фамилия) (подпис)

***Председател на изпитната комисия**:..... .....*

(име, фамилия) (подпис)

***Директор/ръководител на обучаващата институция**:.....*

(име, фамилия) (подпис)

(печат на училището/обучаващата институция)