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: