**СУ „Св. Климент Охридски“, ФМИ**

Специалност „Софтуерно Инженерство“

**Обектно-ориентирано програмиране, 2019-2020 г.**

**Задача за домашно № 1**

**Спазвайте практиките за обектно-ориентирано програмиране, коментирани на упражнения и лекции.**

## **Задача 1 (3 точки)**

*Може да се използва <cstring>*

Напишете собствена имплементация на класа String, която да бъде с име MyString. MyString трябва да може да съдържа текст и да поддържа инициализиране с низ, добавяне на символ в края на низа, принтиране, както и оператори за сравнение (==, !=, >, <) (подобна на strcmp)

Примерен main():

|  |
| --- |
| MyString temp("Hello world"); temp.print(); temp.append('!'); temp.print(); |

Примерен изход:

|  |
| --- |
| Hello world Hello world! |

## **Задача 2 (3 точки)**

Напишете клас IntCounter, който ще съхранява динамично зададена int променлива. IntCounter има следните свойства:

* Съдържа броя на "референциите" към тази променлива (всяка различна променлива, има различен брояч)
* При инициализация, приема указател - стойността, която ще се пази
* При създаване на копие на IntCounter, се увеличава броя на референциите
* Метод, който връща стойността на пазената променлива  
  + При унищожаване на IntCounter, брояча се намалява. Ако брояча стигне до 0, то стойността която се пази, се унищожава

Примерен main():

|  |
| --- |
| int\* some\_number = new int(5); IntCounter first(some\_number); std::cout << first.get\_count() << std::endl; IntCounter second = first; std::cout << first.get\_count() << std::endl;  {  IntCounter third(second);  std::cout << first.get\_count() << std::endl; } std::cout << first.get\_count() << std::endl; |

Примерен изход:

|  |
| --- |
| 1 2 3 2 |

Примерен main():

|  |
| --- |
| int\* some\_number = new int(5); IntCounter first(some\_number);   int\* some\_other\_number = new int(7); IntCounter second(some\_other\_number);  std::cout << first.get\_count() << std::endl; std::cout << second.get\_count() << std::endl; |

Примерен изход:

|  |
| --- |
| 1 1 |

## **Задача 3 (4 точки)**

Моделирайте опростена версия на SVG (Scalable Vector Graphics), която да поддържа визуализация на правоъгълници, дефинирани чрез две точки (лежащи на двата края на диагонала на правоъгълника), в двумерното пространство. SVG да има следните функционалности:

* Добавяне на нов правоъгълник
* Създаване на правоъгълник по две точки и добавяне към списък с правоъгълници
* Извеждане индекса на правоъгълника, който има най-голямо лице, сред всички правоъгълници от гореспоменатия списък.

Ограничения и изисквания

* Предаване на домашното в указания срок от всеки студент като .zip архив със следното име: **(номер\_на\_домашно)**\_SI\_**(курс)**\_**(група)**\_**(факултетен\_номер)**, където:
  + **(номер\_на\_домашно)** е цяло число, отговарящо на номера на домашното за което е отнася решението (например 1);
  + **(курс)** е цяло число, отговарящо на курс (например 1);
  + **(група)** е цяло число, отговарящо на групата Ви (например 1);
  + **(факултетен\_номер)** e цяло число, отговарящо на факултетния Ви номер (например 63666);
* Архивът да съдържа само изходен код (.cpp и .h файлове) с решение отговарящо на условията на задачите, като файловете изходен код за всяка задача трябва да са разположени в папка с име (номер\_на\_задача), където (номер\_на\_задача) е номера на задачата към която се отнася решението;
* Не е разрешено да ползвате класове от библиотеката STLкато std::string, std::vector, std::stack и др.
* Качване на архива на посоченото място в Moodle;

Пример за .zip архив за домашно: 1\_SI\_1\_1\_63666.zip