Обектно-ориентирано програмиране
Домашна работа №2

Пояснение:

- Реализирайте задачите, спазвайки добрите ООП практики (валидация на данните, подходяща капсулация и т.н.).
- Решения, в които не са спазени ООП принципите, ще бъдат оценени с 0 точки.
- Предадените от вас решения трябва да могат да се компилират успешно на Visual C++ или GCC.
- **Не е разрешено** да ползвате библиотеки от STL и STL функции.

Изисквания за предаване:

- Всички задачи ще бъдат проверени автоматично за преписване. Файловете с голямо съвпадение ще бъдат проверени ръчно и при установено плагиатство ще бъдат анулирани.
- Предаване на домашното в указания срок от всеки студент като .zip архив със следното име:

  (номер_на_домашно)_SI_(курс)_(група)_(факултетен_номер)

  - (номер_на_домашно) е цяло число, отговарящо на номера на домашното, за което се отнася решението (например 2);
  - (курс) е цяло число, отговарящо на курса Ви (например 1);
  - (група) е цяло число, отговарящо на **административната Ви група** (например 1);
  - (факултетен_номер) е низ, отговарящ на факултетния Ви номер (например 12345 или 1MI01234);

Пример за .zip архив на текущото домашно: 2_SI_1_1_12345.zip

Архивът да съдържа само изходен код (.cpp и .h/.hpp файлове) с решение, отговарящо на условията на задачите, като файловете с изходен код за всяка задача трябва да са разположени в папка с име (номер_на_задача).

Качването на архива става на посоченото място в Moodle.

Обектно-ориентирано програмиране
Домашна работа №2

Задача 1. MultiSet

Напишете клас **MultiSet**, който съдържа мултимножество от числа в диапазона от 0 до n, където n е подадено в конструктора. В конструктора се подават 2 числа: **n** (най-голямото число в множеството) и **k** - колко най-много бита са необходими за запазването на броя срещания на едно число (1 <= k <= 8).
Това означава, че всяко число може да се среща в множеството най-много 2^k-1 пъти.
Класът ви трябва да е **максимално оптимален откъм памет**.

Трябва да поддържате следните функционалности:

- Добавяне на число.
- Проверка колко пъти се съдържа число.
- Принтиране на всички числа, които се съдържат в мултимножеството.
- Принтиране на това как мултимножеството е представено в паметта.
- Сериализация/десериализация в/от двоичен файл.
- Сечение/Разлика на две мултимножества.
- Допълнение на мултимножество *(ако x се е срещал p пъти, то в допълнението се среща 2^k - 1 - p пъти).*

Задача 2. ModifiableIntegersFunction

Напишете клас **ModifiableIntegersFunction**, който приема указател към функция и позволява модификации върху нея. Функцията преобразува 16-битови числа (приема 16-битово знаково число и връща 16-битово знаково число).

Трябва да поддържате следните модификации:

- **Задаване на резултат за конкретен вход**: Това означава, че може да определите специфичен изход за даден вход. *(Например, ако имате функцията f(x) = x * 2, може да зададете, че за x = 3, вместо 6, функцията трябва да връща 10).*
- **Изключване на точка**: Това позволява да направите функцията "частична", което означава, че за определени входни стойности функцията няма да върне резултат.
  (*В примера с f(x) = x * 2 може да изключите стойността x = 3, така че при опит за изчисление на f(3) да се получи грешка или специален сигнал, че функцията за този вход не е дефинирана.)*

Обектно-ориентирано програмиране
Домашна работа №2

Трябва да се предефинират оператори за:

- Събиране/изваждане на функции
- Композиция на функции

(Ако за дадена точка някоя от функциите не е дефинирана, то и резултатната не е дефинирана).

- Оператори за сравнение между функции, които оценяват изходите от двете функции при всички възможни входове. (f < g ⇔ f(x) < g(x) за всяко x)
  Ако една функция не дава резултат за определен вход, този случай се третира като имащ най-ниска стойност при сравнението.
- Проверка дали графиките на две функции са успоредни.
- Оператор ^ за многократно приложение (f^k(x) = f(f(f…f(f(x))...)))
- Генериране на обратната (f^-1) функция, ако функцията е обратима
  f^-1 (x) е дефинирана ⇔ f(x) е дефинирана.

Имплементирайте и функции за:

- Проверка дали функцията е инекция/сюрекция/биекция.
- Сериализация/десериализация в/от двоичен файл.
- Изчертаване на функцията в дадена част от равнината [x1 … x2] // [y1… y2], където x2 - x1 = 20 и y2 - y1 = 20.