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)_IS_(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_IS_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.

ЗАДАЧА: Разглеждаме абстрактен базов клас ***Частична Функция***, който преобразува цели 32-битови числа в цели 32-битови числа и задължително притежава операция за проверка дали функцията е дефинирана за дадена точка и операция за пресмятане на резултата на функцията за подадено x.

Да се реализират следните конкретни наследници на абстрактния базов клас *Частична функция*:

- ***Частична функция по критерий*** – в конструктора се подава *функция (това е функция или обект, който се държи като такава)*, която по подадено ѝ като аргумент число, връща наредена двойка - дали функцията е дефинирана там и ако да, какъв е резултатът.
- ***Максимум на частични функции*** – в конструктора се подават няколко *Частични функции* и новосъздаденият обект (максимума на подадените *Частични функции*) е дефиниран в дадена точка, само ако всички подадени функции са дефинирани в нея. Резултатът за дадено x ще бъде максимумът от резултатите на подадените функции за същото x.
- ***Минимум на частични функции*** — в конструктора се подават няколко *Частични функции* и създаденият обект (минимума на подадените *Частични функции*) отново е дефиниран в дадена точка, само ако всички подадени функции са дефинирани в нея. Резултатът за дадено x ще бъде минимума от резултатите на подадените функции за същото x.

Да се реализира програма, която прочита от двоичен файл **func.dat** информация за частична функция и конструира нова частична функция съгласно указаните в двоичния файл правила.

В началото на двоичния файл има две цели неотрицателни 16-битови числа **N** и **T**, за които е изпълнено следното:

- стойността на N не надхвърля 32
- стойността на T определя съдържанието на двоичния файл по-нататък и как се конструира съответната функция. Възможните стойности за T и правилата за конструиране на нова частична функция, стоящи зад тях, са следните:
    - **0** – следват 2N цели 32-битови числа, които определят функцията (<arg1>... <argN> <res1> ... <resN>). Функцията е дефинирана **само** в подадените аргументи.
    - **1** – следват N цели 32-битови числа, определящи частична функция, която не е дефинирана **в нито едно** от дадените числа. За всяко друго подадено x функция да връща x.
    - **2** – следват N цели 32-битови числа, определящи частична функция, която връща 1, само ако като аргумент е подадено някое от тези числа, и 0 за всяко друго. Функцията е дефинирана **за всяко** число.

- **3** – следват N низа, всеки от тях терминиран с 0 и описващ път към двоичен файл. Подадените двоични файлове също задават частични функции, като техният **максимум** представя текущата частична функция.
- **4** – следват N низа, всеки от тях терминиран с 0 и описващ път към двоичен файл. Подадените двоични файлове също задават частични функции, като техният **минимум** представя текущата частична функция.

Програмата да работи в два режима:

1. Приема от стандартния вход две цели числа **a** и **b** и извежда резултатите от изпълнението на функцията за всички числа в интервала [**a**; **b**].
2. Позволява последователно генериране на резултат за всяка дефинирана точка, като всеки следващ елемент се генерира при поискване от потребителя.

Да се обработват по подходящ начин различните грешки, свързани с некоректен вход.

Пример:

func.dat	**first.dat**	**second.dat**	**third.dat**
3 3 first.dat second.dat third.dat	7 0 0 1 2 3 5 6 7 0 3 3 3 4 4 0	2 1 3 5	4 2 0 5 6 7

При въведени числа **a** = 0 и **b** = 10, се очаква да се изведе:

f(0) = 1 f(1) = 3 f(2) = 3 f(6) = 6 f(7) = 7

Забележка: Съдържанието на двоичните файлове е показано като текст само за удобство на примера. **Файловете трябва да са двоични!**