I. **Задача 1. А)** Реализирайте абстрактен клас Question, който представя въпрос от тест (до 100 знака) и поддържа методите void askQuestion(), който извежда въпроса на екрана, въвежда отговор от потребителя и го запазва и bool checkQuestion(), който връща дали отговора на въпроса е въведен и верен. Реализирайте и конкретните наследници IntegerQuestion, който очаква като отговор цяло число, като верният отговор се задава в конструктора и OpenQuestion, който очаква като отговор произволен текст (до 255 знака), като верността на отговора се въвежда от екзаминатор, преглеждащ въпроса и дадения отговор.

Б) Да се реализира клас Test, представляващ списък от въпроси с произволна дължина. За класа да се реализират подходящ конструктор, метод void performTest(), задаващ последователно въпросите на потребителя и int numCorrectAnswers(), връщащ броя на верните отговори.

Задача 2. Да се дефинира шаблон на клас Relation<T>, който съдържа два обекта от тип T, наречени subject и object, и низ с произволна дължина relation, описващ връзката между тези обекти.
Пример: Relation<int> r1(2,6,“is smaller than”),r2(6,3,“is divisible by”);
За шаблона да се реализират голямата четворка и операция за отпечатване void print()
Пример: r1.print(): *2 is smaller than 6.*
За инстанцията на шаблона Relation<int> реализирайте и оператор за композиция * по следния начин.
Ако r = r1 * r2, то r.subject = r1.subject, r.object = r2.object,
r.action = r1.action r1.object “, which is“ r2.action
Пример: (r1*r2).print():*2 is smaller than 6, which is divisible by 3*
Композицията се допуска само ако r1.object == r2.subject, в противен случай резултатът е r1.
Упътване: Обръщението към ф-ята itoa(val,str,10) записва числото val в низа сочен от str.

Задача 3. Да се реализира абстрактен базов клас IntSet, който описва следните операции върху изброимо крайно множество от цели числа:

- bool member(int x): проверява дали цялото число x е елемент на множество.
- int get(int i): връща i-тия елемент на множество. Индексацията на елементите е без значение.
- bool operator < ([попълнете правилния тип] s): проверява дали дадено множество е същинско подмножество на s.
- bool operator * ([попълнете правилния тип] s): проверява дали две множества имат непразно сечение.

Да се реализират наследници IntRange и ArraySet. Клас IntRange представя затворен интервал от цели числа. Краищата на интервала да се задават при конструиране на обекта. ArraySet е клас, поддържащ множество от максимум n цели числа, където n се задава по време на конструиране на обекта. Класът да поддържа следните операции:

- bool insert(int x): добавя числото x към множеството. Ако капацитетът е изчерпан, връща лъжа. Връща истина в противен слуай.
- bool remove(int x): премахва числото x от множеството. Ако елементът не съществува, резултатът е лъжа. Резултатът е истина в противен случай.

Да се реализира функция bool mon([попълнете правилния тип] sets[], int n), която получава масив от обекти (или указатели към обекти) множества. Масивът е с големина n. Функцията да проверява дали елементите на масива образуват строго монотонно растяща редица.

I. Да се реализира абстрактен клас **Task**, описващ задача. Всяка задача да поддържа следните операции:
   1) bool finished () – приключила ли е задачата;
   2) void doWork () – извършване на работа по задачата;
   3) bool canStart () – може ли да започне работа по задачата в текущия момент;
   4) void print () – отпечатване на информация за задачата.

II. Да се напишат две реализации на **Task** – **StepTask** и **DependantTask**.
   1) Клас **StepTask** реализира задача, която приключва след определен брой извършвания на работа по нея (брой извиквания на метода doWork). Необходимият броя стъпки се задава при конструиране на задачата. При отпечатване се предоставя информация за името и номера на задачата.
   2) Клас **DependantTask** реализира задача, която приключва за една единствена стъпка (извикване на метода doWork), но не може да започне, преди друга определена задача да е приключила. Блокиращата задача се подава при конструиране на зависимата задача. При отпечатване се предоставя информация за името и номера на задачата и информацията за блокиращата задача. Обърнете внимание, че цикли са невъзможни!

III. Да се реализира функция void completeAll (Task *[], int n), която по масив от задачи довежда всички до приключило състояние. Функцията итеративно изпълнява стъпки от всички задачи, които могат да бъдат започнати в дадения момент. Процесът продължава до тогава, докато броят на задачите, които могат да започнат, стане 0. Ако в този момент има задачи, които не са приключили да се отпечата информация за тях.