Computer Programming 2 Midterm Exam-
April 4th 2022

- You have 2 hours to complete the assignment.
- If the code does not compile, the exercise won't be accepted for submission.
- Tests are provided to check if your code is correct. You cannot modify the tests!
- If a test does not pass, that exercise won't be accepted for submission.
- Code is expected to be readable, clean, and optimal.
- A skeleton of the exercise is provided. Feel free to add more files and/or include more libraries of you need them.
- To check if your code is correct, you can write the code you need in the main() of the file
main.cpp, but leave it unchanged before submitting the exam.
- Inside the code, replace "TYPE YOUR NAME HERE" with your complete name.
- When you finish, ZIP the whole folder with a filename called "lastname_name.zip" and
upload it to the "Midterm Exam" folder.

Exercise 1 (6 points).

In order to have two different characters to play, you have to implement:

- The abstract class **Character** with the following attributes and methods:
  - A private string called name to store the name.
  - A private int called `healthPoints` to store the health points.
  - Two protectet int called `damage and position;`
  - Public methods:
    - Constructor that given a name, the number of health points, damage, range and position initializes the corresponding attributes.
    - Virtual Destructor.
    - `getName`() that returns the name of the unit.
    - `getHealthPoints`() that returns the life points of the unit.
    - `getPosition`() that returns the position of the unit.

- `recieveDamage`(int damage) that will update the life points of the character according to the damage received. Life points cannot be less than 0.
- `move`(int distance) that will rmove the unit acording to the distance given.
- `upgrade`() that will upgrade the unit. This method must be a pure virtual method.
- `attack`(Unit* other) that will deal damage to the other unit. This method must be a pure virtual method.

- The class **Archer** derived from **Unit** with
  - a private int called range
  - and the following public methods:
    - Constructor that accepts position which calls the base Constructor and initializes the name of the character to “Archer”, the life points to 10, damage to 3 and range to 3
    - Implementation of the abstract function declared in the base class, `upgrade`() increments **range** by 1.
    - Implementation of the abstract function declared in the base class, `attack`(Unit* other) checks if the other unit is within range (distance is less or equal that range). If this is the case, it deals damage to the other unit calling `recieveDamage(`damage`).`
- The class **Spearman** derived from **Unit** with the following public methods:
  - Constructor that accepts position which calls the base Constructor and initializes the name of the character to “Spearman”, the life points to 15, and damage to 6.
  - Implementation of the abstract function declared in the base class, `upgrade`() increments **damage** by 1.
  - Implementation of the abstract function declared in the base class, `attack`(Unit* other) checks if the other unit is an archer, comparing the name. If this is the case it deals damage to the other unit calling `recieveDamage`(damage).

Exercise 2 (4 points). Implement the function:

int ***partialSum***(int **pIntArray*, int *pArraySize*, int *pInitPos*, int *pEndPos*);

which, given the array pIntArray of size pArraySize, it returns the sum of the elements in the range [pInitPos, pEndPos].

For example, if the input array is:

[0, 1, 3, 4]

and *pInitPos* is 1, and *pEndPos* is 2, the returned value will be 4 (that is, 1 + 3).

IMPORTANT!

- You are not allowed to use the operator [ ].
- You have to check for the following errors:
  - The range is valid (pInitPos is smaller or equal than pEndPos).
  - pInitPos and pEndPos are indices within bounds of the array.
- In case of errors, the function must return 0!