Indian Institute of technology, Kharagpur
Software Engineering Lab (CS29006/CS29202)
LabTest 01, Spring 2023
Duration: 1.5 Hours, Marks 30

General Instructions:

1. You will solve these problems in the lab computers. You are supposed to use Gedit as the editor and g++ as the compiler. If you don't have them in the Lab computer, please let us know.
2. No books, copies or electronic devices are allowed to be kept with you. Mobile phones need to be switched off.
3. If you are in need of white sheets, please ask the Tas. They will provide these to you.
4. You are provided with two cpp files "labtest_01_problem_01.cpp" and "labtest_01_problem_02.cpp". All your codes should be written inside these files only. You need to rename these files by appending your roll number at the end. For instance, if your roll number is 21CS10023 then you should submit 2 separate .cpp files named as: "labtest_01_problem_01_21CS10023.cpp" and "labtest_01_problem_02_21CS10023.cpp"

PROBLEM 01: [20 marks]

In this problem you will implement a shape hierarchy starting with an abstract concept shape. As this is an abstract concept, it should be the top of the hierarchy and be implemented as an abstract base class. Four different shapes "Right Triangle", "Rectangle", "Square" and "Circle" inherits from the abstract base class "Shape".

Each type of shape should be able to return its perimeter (double), area (double) and type (string). The entire design should be polymorphic. The child classes should have appropriate data members, constructors/destructors and polymorphic member functions. You are provided with a barebone cpp file "labtest_01_problem_01.cpp". Examine the main() function that implements an iterative menu selection process for users to create Shapes and add them to a list (vector) of Shape *'s. After exiting the menu, the Shapes in the list will be printed out along with their type, perimeter and area. see the menu output and comments to infer what the parameters/data members/member function names should be. You should not touch the code inside main(). Read the comments inside the code carefully to infer where you need to write your codes.

Sample Input and Output: (text in bold font below is not printed in console, they are for your understanding)

Inputs:

1 3 4 ***(Right triangle with b=3, h=4)***
2 3 4 ***(Rectangle with b=3, h=4)***
3 4 ***(Square with side = 4)***
4 2 ***(Circle with radius = 2)***
0 ***(Quit and print)***

Outputs:

Right Triangle: Area=6 Perim=12
Rectangle: Area=12 Perim=14
Square: Area=16 Perim=16
Circle: Area=12.5664 Perim=12.566

PROBLEM 02: [10 marks]

You are tasked with designing a game similar to "The Last of Us." The game involves a player, Ellie, who must survive against a horde of zombies. Each Zombie has a point value; when killed, it adds its point value to the player's score.

You must create a polymorphic design for the different types of zombies that can appear in the game. These include:
Runner: a fast-moving zombie worth 10 points
Stalker: a sneaky zombie worth 20 points
Clicker: a zombie that uses echolocation to detect prey worth 30 points
Bloater: a giant, brutal Zombie worth 50 points

The Player class should keep track of the player's score, while the Zombie class should define the common interface for all zombies. Each subclass of Zombie should implement its own version of getValue(), which returns the point value for that particular type of Zombie.

The player class has a default constructor and two member functions: addToScore and getScore, which add a given value to the player's score and return the current score.

The Zombie class is an abstract base class with two member functions: isDead and getValue. The former returns a boolean value indicating if the Zombie is dead, and the latter returns an integer value representing the point value of the Zombie.

Runner, Stalker, Clicker, and Bloater are derived classes of Zombie. They all have a constructor that takes a boolean value as a parameter to set the isDead status of the Zombie.

You are provided with a cpp file “labtest_01_problem_02.cpp” , which contains the main function that creates an instance of Player, initializes the number of zombies, and creates a pointer array of Zombie objects. It initializes the array with instances of the various Zombie subclasses and loops through the array to add the value of each dead zombie to the player's score using the addToScore member function of the Player class. Finally, it deallocates memory for the Zombie array. You should not touch the code inside main().