CCP6124 Object Oriented Programming and Data Structures
Trimester 2510
Faculty of Computing and Informatics
Multimedia University

Submission Date
30 May 2025, 5pm

Individual Assignment
This is an assignment for a group of one to four students. STRICTLY NO COPYING from other sources except codes given in this course. If detected, all parties involved will get 0 marks. The codes must me your own.

Assignment
In this assignment, you are required to implement a "Robot War" simulator using standard C++. The simulator simulates the warfare of robots in a given battlefield. The rules of war are as follows:

1. The battlefield of the war is a 2-d *m* x *n* matrix (for example an 80x50 battlefield matrix) to be displayed on a screen. You can choose any appropriate battlefield representation using characters and symbols to represent robots, battlefield locations and battlefield boundaries.

2. The simulation is turn-based. If you have 3 robots simulated, robot one performs its action plan first, followed by robot two and robot three before going back to robot one, robot two and so on.

3. The position of a robot is not revealed to other robots (robotPositionX and robotPositionY are private members of every robot class). Each robot can perform a 'look' action to see what are around.

4. A robot can take a move action to one of its 8 neighbouring locations : up, down, left, right, up left, up right, down left and down right.

Up left	up	Up right
left	robot	right
Down left	down	Down left

5. The look(x,y) action will reveal a nine-square area to a robot, centred on (robotsPositionX + x, robotsPositionY + y). Thus a look(0,0) will provide the robot with its immediate neighbourhood (an exhaustive list of places it can visit in its next turn). The look action should reveal whether a location is in the battlefield or not, or whether a location contains an enemy robot. A location can only be occupied by one robot.

6. The robot can perform a fire(x,y) action (keeping in mind that different robot have different fire patterns). This action will will have 70% probability to hit and destroy any robot in the location (robotsPositionX + x, robotsPositionY + y). Make sure that the robot will not fire at (0,0). A robot must not commit suicide.

7. Each robot has a type (which determines the robots strategy) and a name. The simulator will be started with a number of robots in different positions, and it will display the actions of robots until the simulation time is up or until one robot remains, whichever comes first.

8. All robots objects should inherit from a base abstract class called Robot. This Robot class is to be inherited by 4 basic abstract subclasses, namely MovingRobot, ShootingRobotSeeingRobot and ThinkingRobot. These 4 classes should not have any overlapping public member functions (methods) and data members. All robot types mentioned below must be inherited (multiple inheritance) from these 4 classes depending on their capacity to move, shoot, see and thinkl.

9. The initial robot type is GenericRobot. It has the following capabilities.

GenericRobot:
In each turn, this robot can inany order
a. think to decide what to do
b. look(x, y) one time where (x, y) is any 8 neighbouring locations or the location of the robot itself.
c. fire(x, y) one time where (x, y) is any 8 neighbouring locations. It has only 10 shells to fire in a match. If a robot uses up all its shells, it will self destruct.
d. move to one of its 8 neighbouring locations or remain in place.
If it can destroy a robot, it can choose an upgrade.

There are three possible areas to upgrade. They are moving, shooting and seeing.
1. Moving: HideBot or JumpBot.
2. Shooting: GenadeBot, SemiAutoBot or ThirtyShotBot.
3. Seeing: ScoutBot or TrackBot
In each area, a robot can choose only one upgrade. For the next upgrade, the robot must choose from the areas not chosen before. After three upgrades, a robot can be upgraded anymore.

The capabilites of the upgrades are as follow:

HideBot : The robot can hide three times in a match. When the robot hides, it cannot be hit by other robots
JumpBot : The robot can jump to a new location anywhere in the map. It can jump three times in a match.

LongShotBot : The robot can fire up to three unit distance away from its location. It means the robot can fire(x, y) where $x + y <= 3$.

SemiAutoBot : Each shell the robot fires is now considered as three consecutive shots into one location and each shot has a 70% probability to hit and destroy another robot.

ThirtyShotBot : The robot now has a fresh load of 30 shells replacing its current load of shells.

ScoutBot : Instead of look(x, y) The robot can see the entire battlefield for one turn. This ability can be used three times in a match.

TrackBot : The robot can plant a tracker on another robot it can see. The location of the targetted robot will be known to the robot until the end of a match. The robot has three trackers.

10. The initial conditions of a simulation are specified in a text file which contains
The dimensions of the battlefield (*M by N*)
The number of simulation steps. (*steps: T*)
The number of robots. (*robots: R*)
The type, name, and initial position of each robot
(if the initial position is `random`, the robot is placed in a random location)

An example text file looks like this:`

```
M by N : 40 50
steps: 300
robots: 5
GenericRobot  Kidd  3    6
GenericRobot  Jet   12   1
GenericRobot  Alpha 35   20
GenericRobot  Beta  20   37
GenericRobot  Star  random    random
```

11. Each robot has three lives. When a robot is destroyed, it can go into a queue to re-enter into the battlefield three times. Only one robot can re-enter each turn to a random location in the battlefield.

12. The above requirements are the basic requirements of this assignment. Add three more new robot types to make the simulation even more exciting.

Implementation

1. In each turn of a simulation, display the battlefield, the actions and the results of the actions of each robot in that turn. In addition, save the same information into a text file.

2. Classes and functions in the standard C++ libraries can be used.

3. Your solution must employ the OOP concepts you have learnt such as **INHERITANCE, POLYMORPHISM, OPERATOR OVERLOADING** and any number of C++ object oriented features.

Deliverables

a) Source code in one file (For example `TC01.1171777777.Tony.Gaddis.cpp`).
b) Design documents such as class diagrams, flowcharts, pseudo codes in PDF format to explain your work.
c) Screen-shots and explanation of your program running compiled into a document in PDF format.

Additional Info on Deliverables

a) Source codes have to be properly formatted and documented with comments and operation contracts. Do not submit executable files.

b) For ALL your .cpp and .h files, insert the following information at the top of the file:

```
/**********|**********|**********|
Program: YOUR_FILENAME.cpp / YOUR_FILENAME.h
Course: Data Structures and Algorithms
Trimester: 2410
Name: Frank Carrano
ID: 1071001234
Lecture Section: TC101
Tutorial Section: TT1L
Email: abc123@yourmail.com
Phone: 018-1234567
**********|**********|**********/
```

Soft-copy submission instruction

a) Create a folder in the following format:

TUTORIALSECTION_ASSIGNMENT_FULLNAME

For example, if your name is Frank Carrano, you come from TC201 tutorial section, and you are submitting Milestone 1, then your folder name should be "TC201_M1_FRANK_CARRANO" without the double quotes.

b) Place all files for submission into the folder.

c) Zip your folder to create a zip (TC201_A1_FRANK_CARRANO.zip) archive. Remember that for Milestone 2, your zip archive filename should be "TC201_M2_FRANK_CARRANO.zip".

d) Submit your assignment through MMLS.

Evaluation Criteria

Criteria	Max
Design documentation (must include a class diagram	5
Initialization of a simulation	3
Display and logging of the the status of the battlefield at each turn	5
Display and logging of the actions and the status of each robot at each turn	3
Implementation of the required robot classes with OOP concepts	10
The algorithms used to optimize the actions of robots listed in the assignment document	5
Implementation of three new robot classes (3 marks for each robot)	9
Total	40

Each feature will be evaluated based on fulfilment of requirements, correctness, compilation without warnings and errors, error free during runtime, basic error handling, quality of comments, user friendliness, and good coding format and style.