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

Submission Date
Week 14: 07 Friday 2024, Friday, 11:59 pm

Individual Assignment
This is an assignment for a group of two to four students. Individual students are not allowed to work alone. All members of the group must be from the same tutorial section. STRICTLY NO COPYING from other sources except codes given in this course. If detected, all parties involved will get 0 marks. The codes must be your own.

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

1. The simulator will read the details of the battle from a text file that contains the types of ships, the number of each ship type, and the naval battlefield (2-d *m* x *n* matrix). The text file has the following format:

game.txt
```
--------------------------------
iterations 100
width 10
height 10
Team A 4
Battleship  *  5
Cruiser  $   4
Destroyer  #   4
Frigate  @    3
Team B 2
Battleship  <  2
Cruiser  >   3

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 0 1 0 0

Line content	meaning	Extra details
iterations 100	Number of turns to run the simulation	Running all the ships 1 time is 1 turn
width 10	width of the naval battlefield	10
height 10	height of the naval battlefield	10

Line	Type of ship	Notes
Team A 4		Team's name is A and has 4 types of ships
Battleship * 5	Battleship	* is the symbol to use in the display, *1 is the first battleship, *2 is the second, …, *5 is the last battleship.
Cruiser $ 4	Cruiser	$1 $2, $3, $4
Destroyer # 4	Destroyer	#1 #2, #3, #4
Frigate @ 3	Frigate	@1, @2, @3

Line	Type of ship	Notes
Team B 2		Team's name is B and has 2 types of ships
Battleship > 2	Battleship	> is the symbol to use in the display, >1 for the first battleship and >2 for the second.
Cruiser < 3	Cruiser	<1 and <2

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0	0 is location available for ships placement 1 is a part of an island

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0	

2. The naval battlefield of the war is a 2-d *m* x *n* matrix to be displayed on a screen.

3. Battleships will be created as objects and are placed at random positions on the naval battlefield's locations with 0 value.

4. The simulation is turn-based. If you have 3 ships simulated, ship one performs its action plan first, followed by ship two and ship three before going back to ship one, ship two and so on. In each turn, depending on the capabilities of a ship, the ship will first perform a "look" action (if the ship has that capability), then make a "move" (if the ship has that capability), followed by a "shoot" or "destroy" action (if the ship has that capability too).

5. The position of a ship (shipPositionX and shipPositionY) is known to all the ships in the same team and not known for the other ships in the opposite team. In general, each ship can perform a 'look' action to see what are around.

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

	up	
left	ship	right
	down	

7. A ship can take a **shoot** 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	ship	right
Down left	down	Down left

8. A ship can take a **destroy (take the enemy ships' place)** action to one of its 4 neighbouring locations: *up, down, left, and right.*

	up	
left	ship	right
	down	

9. The ***look(x,y)*** action will reveal a nine-square area to a ship, centred on *(shipPositionX + x, shipPositionY + y).* Thus a *look(0,0)* will provide the ship 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 ship. A location can only be occupied by one ship.

10. The ship can perform a ***shoot(x,y)*** action (keeping in mind that different ships have different shooting patterns). This action will exterminate any ship in the location *(shipPositionX + x, shipPositionY + y).* Make sure that the ships will not shoot at (*0,0). A ship must not commit s*uicide. A ship must not shoot any team members.

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

12. All ships' objects should inherit from a base abstract class called Ship. This Ship class is to be inherited by 4 basic abstract subclasses, namely MovingShip (move function), ShootingShip (shoot function), SeeingRobot (see function) and RamShip (stepping function). These 4 classes should not have any overlapping public member functions (methods) and data members. All ships' types mentioned below must be inherited (multiple inheritance) from these 4 classes depending on their capacity to look, shoot, move or destroying on other ships.

13. The initial ship types are defined as follows:

Battleship: In each turn, this Battleship looks at its current position and decides on a move. Then moves (once) and shoots (2 times) at random positions. This sequence is repeated for each turn. The maximum city block distance of firing should be set to 5 (i.e. for any *fire(x,y), x+y* is at most 5). Battleship cannot move to a location that contains another ship. If Battleship shoots a total of 4 Ships, then it will be upgraded to be allowed to move to a location that contains another ship, thus destroying and killing that ship. This type of ship is called **Destroyer** (see below)

Cruiser: In each turn, this ship looks at the immediate neighbourhood (3x3 window centred at the ship's position) and performs a move to one of the neighbours. Thus, a **Cruiser** will not shoot at all and will terminate ships in its path. It will prefer locations which contain enemy ships. If the Cruiser kills a total of 3 other ships, then the **Cruiser** will be upgraded to be allowed to shoot in a similar way to a **Battleship** shoot, this type of ship will now become a **Destroyer** as well.

Destroyer : This is either an upgraded Battleship or an upgraded Cruiser, it means it is a ship that can shoot like Battleship and destroy other ships like **Cruiser**. If the **Destroyer** kills 3 other ships, it will be upgraded to **SuperShip** (see below).

Frigate: This ship does not move and cannot look. It shoots at only one of its immediate neighbouring cells surrounding it. In each turn, the target location to fire is not random. The sequence of firing is such that it starts from the location up and continue with a new location clockwise in each following turn. If Frigate shoots a total of 3 other ships, it will be upgraded to **Corvette**.

Corvette: This ship does not move. It shoots at only one of its immediate neighbouring locations surrounding it randomly.

Amphibious: This ship can move in the sea or on islands. **Amphibious** behaves like a Battleship. If Amphibious terminates 4 other ships, it will be upgraded to **SuperShip.**

SuperShip: This ship moves like the **Cruiser**, destroying and killing any ship on its path. On top of that, in each turn, this ship shoots randomly at 3 locations in the battlefield.

14. Each ship has 3 lives. When a ship is destroyed, it can go into a queue to re-enter into the battlefield three times. In each turn 2 ships can re-enter at random locations in the battlefield.

15. The above requirements are the basic requirements of this assignment. You may feel free to add more features and capabilities, and even your own robot types to make the simulation exciting.

16. One of the two teams win when the other team is completely annihilated.

Implementation

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

2. The use of containers provided by standard C++ libraries such as vectors, queues and linked lists is not allowed in this assignment. Write your own codes to implement the data structures you need.

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

4. During a simulation, the simulator needs to know which ship is the next ship to take its action. Select an appropriate data structure to keep track of this.

5. Use queue(s) to keep track of the ships which have been destroyed and are waiting for their turns to re-enter into the battlefield.

Deliverables

a) Source code in one file (For example `TC01.1171777777.Tony.Gaddis.cpp`).
b) Design documents such as class diagrams, flowcharts or 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 ebwise.

Evaluation Criteria

Criteria	Max
Design documentation	5
Initialization of a simulation	1
Display and logging of the the status of the battlefield at each turn	3
Display and logging of the actions and the status of each ship at each turn	3
Implementation of the required ship classes with OOP concepts	8
The algorithms used to optimize the actions of ships	5
Design and implementation of data structures (linked list and queues)	5
Extra	
Implementation of additional ship classes (have to be fundamentally different than the one provided)	3
Interview	
Slide design	2
Presentation skills	2
Questions Answering (3 questions each 2 marks)	6
Total	43

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.