National University of Computer and Emerging Sciences

FAST School of Computing **Spring 2020** **Islamabad Campus**

CL118:Programming Fundamental Lab

Remote Final Exam

Attempt Time: 3 Hours
Submission (on google classroom and through email) Time: 15 minutes

Wednesday, July 15, 2020

Total Marks: 100

Lab Instructor(s)

Mr. Shehreyar Rashid, Muhammad Usman,
Miss Shafaq Riaz & Miss Shahnila rahim

Instructions:

1. The final exam will be attempted offline.
2. Understanding the question paper is also part of the exam, so do not ask any clarification.
3. You must write generic codes that should be able to work on arrays of different sizes.
4. If we unable to compile because of syntax error no marks will awarded.
5. **Submission Instructions**:
   a. Save all .cpp files with your roll no and task number.
   e.g. i190001_Task01.cpp

   b. Now create a new folder with name **Full course code - Roll number** e.g. CL118-19i-0001
   c. Move all your .cpp files to this newly created directory and compress it into .zip file (do not make .rar file). It must not contain executable files.
   a. Now you must submit this zipped file on google classroom and **MUST also** email to the email address (of the concerned course/lab instructor) which will be provided. They will be given 15 minutes (after the 3 hours attempt time) for this purpose. All students must use the standard file name format (Full course code - Roll number e.g. CL118-19i-0001). Submissions after 30 minutes may not be accepted. **Try to submit soon after 3 hours of attempt time and do not wait for 15 minutes to be elapsed.**
   b. You MUST check your zip file [**1)** Is it virus free **2)** Is required files present or not.]. The student is solely responsible to check the final zip files for issues like corrupt file, virus in the file, mistakenly exe sent. If we cannot download the file from Google classroom due to any reason it will lead to zero marks in the assignment.
6. For proven cheating/ plagiarism, student will get an F grade even if the student had opted for S/U grade, and the case will be referred to DDC (Department's Disciplinary Committee). Instructors will conduct vivas of randomly selected students or in case of doubt (significantly different attempt as compared to past performance in the course or matching attempt with other students). Plagiarism includes sharing an attempt to other students (copy providing). Students who are not able to satisfactorily answer instructor's questions (based on the exam as well as slightly lateral but related concepts) during viva will also be considered as plagiarism cases.
7. Failure to observe above mentioned instructions will lead to disqualification in Exam.

	Q-1	Q-2	Q-3	Total
Marks	**50**	**20**	**30**	**100**

s

Question 1 [50 Marks]

In this question you will simulate a laboratory experiment as a life game. You have a habitat that contains randomly distributed entities. In our experiment our habitat will be an NxN square place/grid.

Each entity has 5 different gates which can be named from { A,B,C,D,E,F,G,H } set.

The Spreading V can infect an entity if the entity has A or B gate. If the entity has 3 of these gates which are named as A,B,C,D than this entity will die after 14 turns otherwise it will recover and get immune to Spreading V. after 30 turns. Dead entities stay in habitat for five turns and be infectious. They cannot move.

Spreading V can infect from one entity to other if they are closer than 3 units (in square form) see attached example for clarification. An entity becomes infectious after 3 turn.

Your simulation will take the start information from input.txt file which will be on the same folder with you executable. This file include information about habitat constants, entities and their movements in turns.

Your simulation will write an output.txt file which will show the infected and died entities and last situation of the habitat. Also, after each turn you will fill another file "turns.txt". This file shows a brief outcome for each turn.

An example of the input and output files are below: We have a 5x5 habitat and includes 3 entity in this example. İt is a short description so the movements are made just for three steps. Test case should be made for more turns. Since our habitat is small and each entity is close to each other infected entity can easily infect the others. But the infection will occur after 3 turns so we do not see any infection in three turns. "X" shows infected entities, "O" for healthy entities, "D" for dead ones.

Your application must not include any console input command. Your app directly read the input file, generate output.txt and turns.txt files and close itself silently. If you prefer you can draw turns on the screen but just the file version will be graded.
Your file must include your name and number as comment at the beginning. (File ex: 19I-1001.cpp)

Do not place any other code file or dependencies for your project.

Example
Input.txt

```
size 5
turn_count 3
entity_count 3
entity 1 ACDEF 2x2 infected
entity 2 BDFHC 4x5 normal
entity 3 CDHGF 5x5 normal
turn 1 2x3 4x5 5x4
turn 2 3x3 4x4 5x3
turn 3 3x4 4x3 5x2
```

Output.txt

```
Normal       : 2
Infected     : 1
Dead         : 0
Recovered    : 0
entity 1 3x4 infected
entity 2 4x3 normal
entity 3 5x2 normal
```

Turns.txt

```
Turn 1:
-------
-      -
-  X   -
-      -
-     0-
-    0 -
-------
Turn 2:
-------
-      -
-      -
-  X   -
-    0 -
-  0   -
-------
Turn 3:
-------
-      -
-      -
-    X -
-  0   -
- 0    -
-------
```

Marks distribution:

Task	Marks
Input File reading and parsing	5
Infection Simulation(Entity can be affected if closer than 3 units)	20
Entity Death simulation	10
File Writing(Turns.txt and output.txt)	15
Total	**50**

Question 2 [20 Marks]

Now let's use programming to simulate bunnies! (Why? Because everybody likes bunnies)
In our simulation, the bunny is on a grid (2D array of size 10x10) at some location defined by an X-coordinate and a Y-coordinate. Also, the bunny has some number of energy units measured in carrot sticks. Note that the X-coordinates, Y-coordinates, and the number of carrot sticks are integer values. The Bunny can hop (move) north, south, east, or west.

- When bunny hops to the north, the bunny's Y-coordinate is increased by 1, and the X coordinate remains unchanged.
- When bunny hops to the west, the bunny's X-coordinate is decreased by 1, and the Y-coordinate remains unchanged.
- When bunny hops to the east, then the X-coordinate is increased by 1 and the Y-coordinate is unchanged.
- When bunny hops to the south, then the Y-coordinate decreased by 1and the X-coordinate is unchanged.
- Bunny must hop within the 10x10 grid, if it exceeds print message "Grid limit exceeds Bunny cannot hop".
- If bunny moved to east, it cannot hop towards north.
- Hop to south must be followed by west.
  In case of violation of above rules, your code should stop taking direction from user and display the message (as shown in test cases)

Note that making one hop requires the bunny to eat one carrot stick, and when bunny has eaten all its carrot sticks, the bunny will stop hopping.

- Start your program by getting X and Y locations and carrot sticks possessed by each bunny from user.
- Then write a function Hop (direction, X & Y coordinates And Carrots). The direction parameter is 0 for north, 1 for east, 2 for south, and 3 for west (like a clock face). The hop( ) function should test to make sure that the Bunny has not eaten all of the carrot sticks – if the Bunny still has carrot sticks, the hop( ) function should update coordinates as explained above and print the message "hop". If no carrot stick remains, it should just print the message "The bunny cannot hop".
- Before calling Hop( ) function from main( ), ask user the direction by choosing a value from 0 to 3 and then pass this value to the Hop( ) function. Your code must validate the direction and display error message (if any).
- Write a function DisplayInfo() that should print the Bunny's location and number of remaining carrot sticks.
- Write a function main() to call all these functions.

Note: Variables must not be Global

Test case 1: [2.5 marks]

X-coordinates: 7
y-coordinate: 6
carrot: 2
Output:
Bunny at location 7, 6 with 2 carrots
direction =1
Bunny at location 8, 6 with 1 carrot
direction =1
Bunny at location 9, 6 with 0 carrots
Bunny cannot hop

Test case 2: [2.5 marks]

X-coordinates: 2
y-coordinate: 8
carrot: 5
Output:
Bunny at location 2, 8 with 5 carrots
direction =2
Bunny at location 2, 7 with 4 carrots
direction =3
Bunny at location 1, 6 with 3 carrots
direction =0
Bunny at location 1, 7 with 2 carrots
direction =2
Bunny at location 1, 6 with 1 carrot
direction =3

Grid limit exceeds, Bunny cannot hop

Test case 3: [2.5 marks]

X-coordinates: 1
y-coordinate: 2
carrot: 3
Output:
Bunny at location 1, 2 with 3 carrots
direction =1
Bunny at location 2, 2 with 2 carrots
direction =2
Bunny at location 2, 1 with 1 carrots
direction =3

Bunny at location 1, 1 with 0 carrots
Bunny cannot hop

Test case 4: [2.5 marks]

X-coordinates: 9
y-coordinate: 2
carrot: 4
direction =3
Output:
Bunny at location 9, 2 with 4 carrots
direction =3
Bunny at location 8, 2 with 3 carrots
direction =1
Bunny at location 9, 2 with 2 carrots
direction =1

Grid limit exceeds, Bunny cannot hop

Test case 5: [5 marks]

X-coordinates: 5
y-coordinate: 2
carrot: 4
Output:
Bunny at location 5, 2 with 4 carrots
direction =1
Bunny at location 4, 1 with 3 carrots
direction =0

bunny cannot hop

Test case 6: [5 marks]

X-coordinates: 1
y-coordinate: 2
carrot: 3
Output:
Bunny at location 1, 2 with 3 carrots
direction =1
Bunny at location 2, 2 with 2 carrots
direction =2
Bunny at location 2, 1 with 1 carrot
direction =0

Bunny cannot hop

Question 3 [30 Marks]

Addition of (Very) Large Numbers

In this problem, you have to take two char arrays of numbers (e.g. 1203009954 and 109876201453) from the user and store each char array (number) in a separate integer array. While storing a number in an array you have to keep in mind that each digit of that number should be stored in a separate index (**apply validation check** (e.g. no alphabets or special characters "(+_-)(*&^%$#@!~`'";:,<>./?" Etc) allowed)). You have to add these two numbers and store answer in a third array digit by digit.
Example:

									1	2	0	3	0	0	9	9	5	4

							1	0	9	8	7	6	2	0	1	4	2	3

+

							1	1	1	0	7	9	2	1	1	3	7	7

Test case1: Marks 10
Input Num 1: "120367"
Input Num 2: "1203a7"
Output: "Second number is invalid"

Test case2: Marks 10
Input Num 1: "1233677001257"
Input Num 2: "1002"
Output: "1233677002259"

Test case3: Marks 10
Input Num 1: "109876201453"
Input Num 2: "1203009954"
Output: "111079211407"

Note:

1. *String and built in functions not allowed.*
2. *Use cin.getline for inputs.*
3. *Max size of arrays will be 20.*

~^Good luck^~