NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

Programming Fundamentals - FALL 2023

ASSIGNMENT- 4

Due Date: **Wednesday 22nd November 2023 at 11:59 pm**

Please follow the following submission instructions. Failure to submit according to the format would result in a deduction of marks. Submissions other than Google classroom (e.g., email etc.) will not be accepted. No late submission will be accepted. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Instructions: **Total Marks: 120**

1. Assignments are to be done individually. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. The code you write must be your own and you must have an understanding of each part you code. You are encouraged to get help from the instructional staff through email, on google classroom or individually visiting their respective offices.
2. The OBJECTIVE of this assignment is to offer you practical experience with arrays and loops in C++.
3. Use appropriate data types, operations, and conditional structures for each problem. You cannot use advanced constructs like **Classes, Structures or pointers** for this assignment.
4. No LATE assignments will be accepted.
5. Displayed output should be well mannered and well presented. Use appropriate comments and indentation in your source code.

Honor Policy

This assignment is a learning opportunity that will be evaluated based on your ability. Plagiarism cases will be dealt with strictly. Plagiarism of any kind (copying from others, copying from the internet etc.) is **NOT** allowed. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all the remaining assignments, or even an F grade in the course.

Submission Guidelines

1. Dear students, we will be using auto-grading tools, so failure to submit according to the below format would result in zero marks in the relevant evaluation instrument.

2. For each question in your assignment, make a separate cpp file e.g., for question 1, make q1.cpp and so on. Each file that you submit must contain your name, student-id, and assignment # on top of the file in the comments.
3. Combine all your work in one folder. The folder must contain only .cpp files (no binaries, no exe files etc.).
4. Rename the folder as ROLL-NUM_PROGRAM_SECTION (e.g., 23i 0001_AI/DS_B) and compress the folder as a zip file. (e.g., 23i-0001_AI_B.zip).
5. Submit the .zip file on Google Classroom within the deadline.
6. 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. **Note: Start early so that you can finish it on time.**
7. I think by now you all know that input validation is a must in every question.

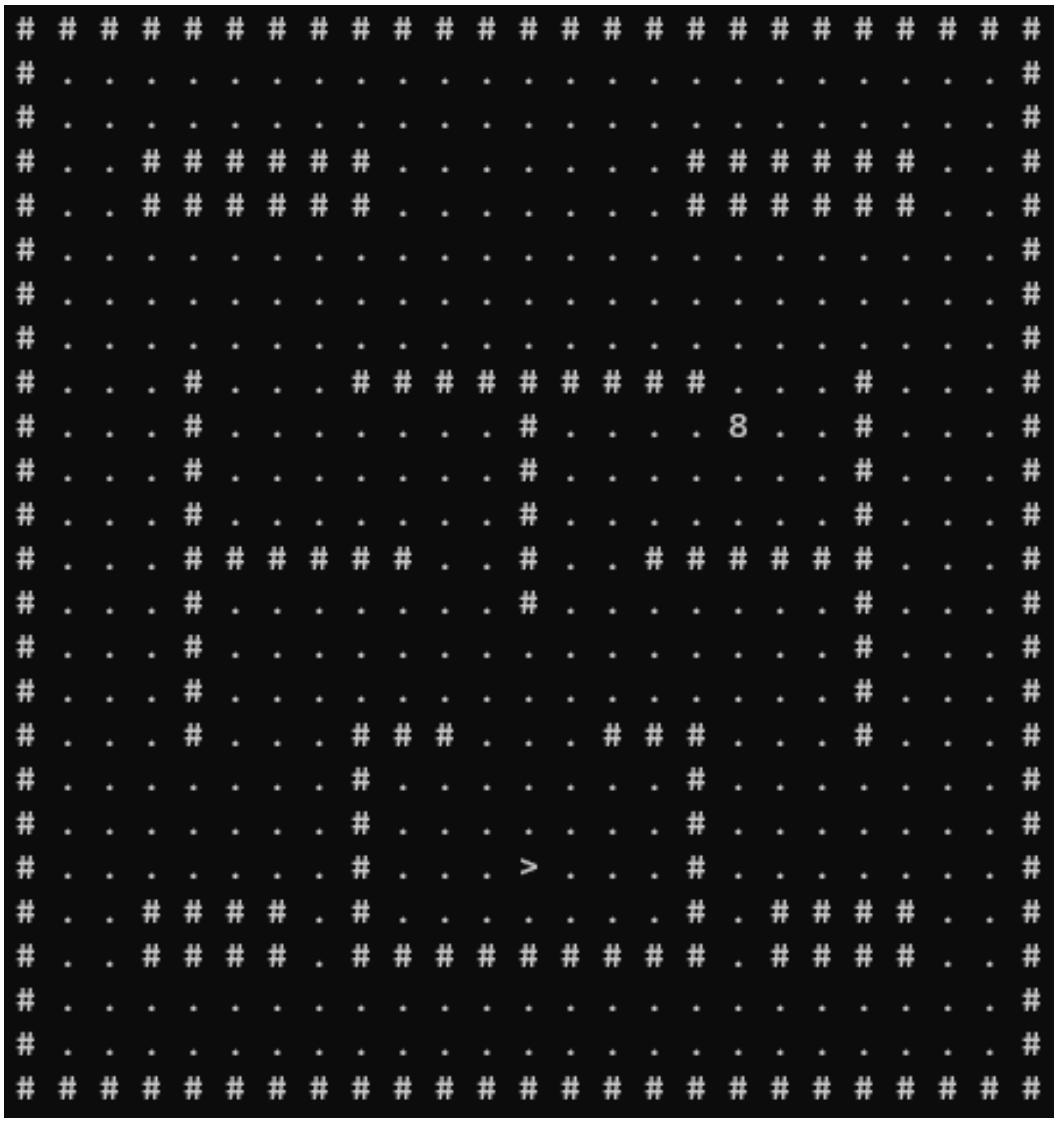
Question 1: Game Development **(35 marks)**

Develop a Pac-man Game. You are not allowed to use pointers for this question, and the game should **ONLY** be developed using the concepts you have learned so far during the course. Rules for creating the game are given below:

1. You will be creating a stage where Pac-man moves according to the user input. Stage is just like a maze representation where Pac-man and Ghost move around but don't cross the boundary of the maze.
2. Ghosts are enemies that chase Pac-man in the maze and the game ends when they make contact with Pan-man.
3. Game must end when all the Pellets are collected.
4. Pac-man would be represented as >, <, V, /\ depending upon user input for left, right, up and down movement.
5. The Pellets must be represented as dots (.).
6. Ghost should look like 8.
7. The Ghost should move randomly in the maze.
8. Walls should be represented with # and the walls have to be hard coded.
9. **You have to create a hard coded 25 x 25 (Two Dimensional) maze.**

Given below is the representation of the maze

Example maze

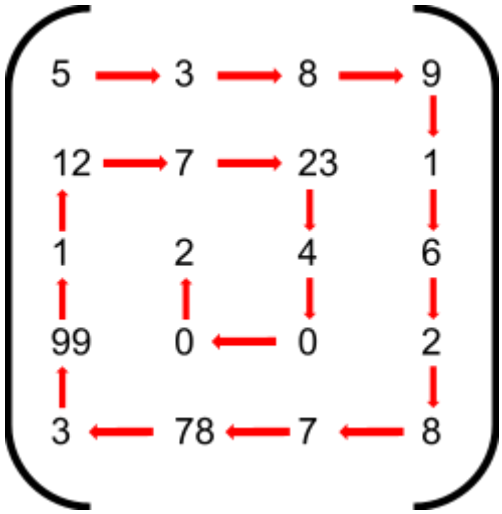


Hint: You can hard code the coordinates of the Pac-man and the ghost and work around that.

Question 2: Custom Matrix Printing Pattern **(15 marks)**

Write a C++ program to print the values of the 2D matrix in the following pattern:

Example



For the above matrix, the printed values should be:

Output: 5 ⟶ 3 ⟶ 8 ⟶ 9 ⟶ 1 ⟶ 6 ⟶ 2 ⟶ 8 ⟶ 7 ⟶ 78 ⟶ 3 ⟶ 99 ⟶ 1 ⟶ 12 ⟶ 7 ⟶ 23 ⟶ 4 ⟶ 0 ⟶ 0 ⟶ 2

Note: The matrix should be randomly populated and your code should work for any square matrix of size n x n: **NOT HARD CODED**

Question 3: 3D Array Quadrant Manipulation **(35 marks)**

Overview:

Write a program that accepts a three-dimensional array. The depth is fixed at 3, and each 2D layer must be a minimum of 8 x 8. Allowable matrix sizes include (3, 8, 8), (3, 10, 10), (3, 16, 16), and so on. Initialize the values of the first layer to zeros, the second layer to ones, and the third layer to twos. The final layer will also serve as the resultant array.

Operations:

Now conceptually divide the last layer into four quadrants. Generate N random numbers for each quadrant, where N is calculated as (elements in quadrant divided by 8). The range of the random numbers for each quadrant should fall within the respective quadrant's indexes. Perform the following operations on each quadrant using the randomly generated indexes:

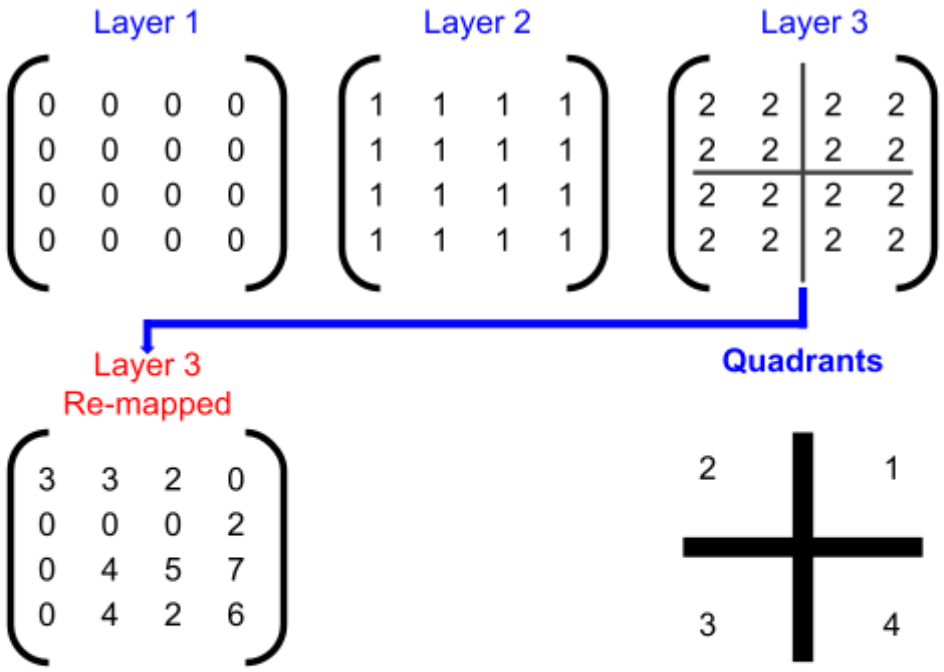
First Quadrant: Replace the value of the last layer at each randomly generated index with the value from the first layer at that index.

Second Quadrant: Update the value of the last layer at each randomly generated index by adding the value from the second layer at that index.

Third Quadrant: Update the value of the last layer at each randomly generated index by adding the existing value from the same layer at that index.

Fourth Quadrant: Compare each index within and across the first three quadrants of the last layer. If there are no matches (i.e., if all values are different), replace the value at that index in the fourth quadrant with the sum of the three different values. If all the three values are not different (i.e., if any two values match or all three match), retain the original value at that index in the fourth quadrant, which is the base value of 2.

Example: This is an example of (3, 4, 4). Your size of each layer should be ≥ **8 x 8**

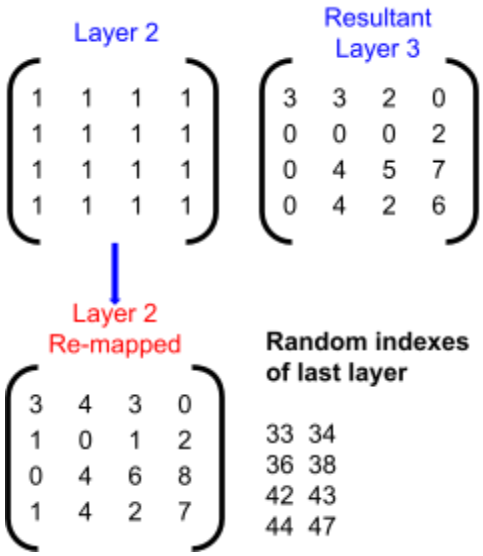


Random indexes for each quadrant

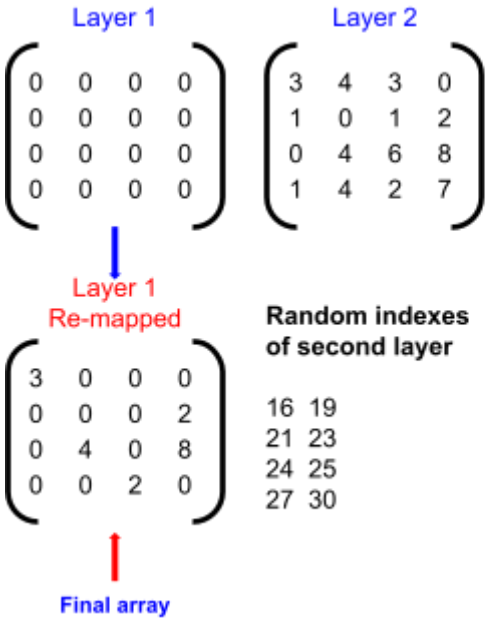
First: 35, 38
Second: 32, 33
Third: 41, 45
Fourth: 46, 47

Sheeda's theorem

Final:
After final layer operations, shift N elements (where N = total last layer elements divided by 8) to the second layer. Select N random indexes, transfer corresponding values to the second layer by adding them to existing values at those indexes. Update the second layer with these new values. Below is the visual representation:



Next, generate N random numbers again, this time calculated as (number of elements in the second layer divided by 8). Take these randomly generated indexes from the second layer and replace the zeroes in the first layer with the values at those indexes.



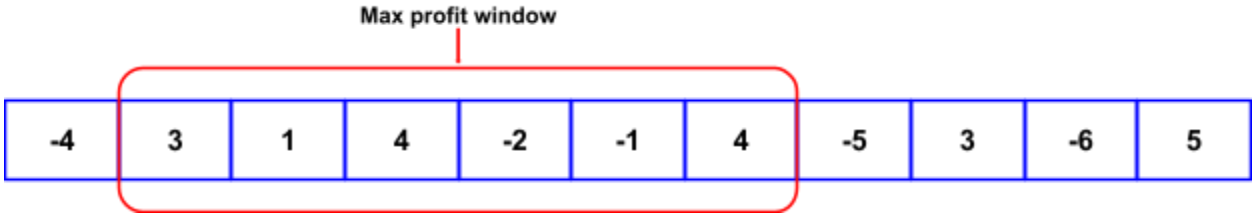
Bonus: Print the final array in ascending order (use an efficient algorithm: don't hard code). For the above given example: Output: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 3, 4, 8

Question 4: Profit Maximization **(20 marks)**

Suppose you are a highly intelligent individual (even though you are not: jk) and you actively participate in buying and selling of cryptocurrencies. One day, you suddenly realize that you have been blessed with a superpower: the ability to accurately predict the gain and loss ratio of Bitcoin for the next two months (60 days). Now, since you know what the gain and loss ratios of Bitcoin will be for the next two months, what you need to do is, you need to buy and sell bitcoin **ONCE** in the span of the next two months. Your **GOAL** is to **MAXIMIZE** the profit earned.

Note: Create an array of size 60 and populate them randomly with integers ranging from -10 to 10. These numbers represent the daily gain and loss with respect to the prior day's price. At the end, display the profit, the start index and the end index of the window.

Example



The above visualization is for your understanding

Question 5: A Series Approximation for Pi **(15 marks)**

Overview:

We all are familiar with the renowned mathematical constant π**,** with a value of 3.141592653… and so forth. Various methods exist for approximating π**.** One such approach involves simulating the act of throwing darts at a circular dartboard. Another method entails partitioning a unit circle into millions of equally-sized sectors, rearranging them into a rectangular formation, and then comparing the formulas for the area of the rectangle and the circle. Nevertheless, there are more reliable methods for calculating the value of π, such as the remarkable mathematical relationship that allows for its computation:

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots$$

$$= \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1}$$

The expression to the right of the equal sign represents an infinite series, where each fraction within the series is referred to as a *term*. By following the pattern of starting with 1, subtracting one-third, adding one-fifth, and continuing with the odd integers, the resulting sum converges towards the value of π/4

Write a program that employs the mentioned formula to compute an approximation of π. Carry out the computation until the value of a *term* becomes less than a defined threshold, specified as a named constant:

constexpr double THRESHOLD = .000001;

To improve the clarity of this calculation, your program should:

1. Include a loop to monitor the running total in the formula at each step.
2. Keep track of the current term by computing the reciprocal of the next odd number.
3. Iterate through the loop until the current term falls below the **THRESHOLD.**
4. Alternate between adding and subtracting each term from the running total.
5. Multiply the total by 4 at the end of the loop.
6. Display the result in a line format like this:

Pi is approximately 3.141590653589692

****** Good Luck ******