Object Oriented Programming

FAST-NU, Islamabad, Summer 2023

Assignment 1

Total marks: 110

Submission Information

Submission Deadline: Sunday **July 2nd, 2023 at 11:30 AM**. You are supposed to submit your assignment on GOOGLE CLASSROOM (CLASSROOM TAB not lab). Only ".ZIP" files are acceptable. Other formats should be directly given ZERO. Correct and timely submission of the assignment is the responsibility of every student, hence no relaxation will be given to anyone.

Tips: For timely completion of the assignment, start as early as possible.

Plagiarism: Plagiarism is not allowed. If found plagiarized, you will be awarded zero marks in the assignment (copying from the internet is the easiest way to be caught).

Instructions:

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

I. For each question in your assignment, make a separate cpp file e.g. for question 1, make ROLL-NUM_SECTION_Q#.cpp (22i-0001_A_Q1.cpp) and so on. Each file that you submit must contain your name, student-id, and assignment # on top of the file in comments.
II. Combine all your work in one folder. The folder must contain only .cpp files (no binaries, no exe files etc.).
III. Run and test your program on a lab machine before submission.
IV. Rename the folder as ROLL-NUM_SECTION (e.g. 22i-0001_A) and compress the folder as a zip file. (e.g. 22i-0001_A.zip). Do not submit .rar file.
V. Submit the .zip file on Google Classroom within the specified deadline.
VI. Submission other than Google classroom (e.g. email etc.) will not be accepted.
VII. 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.
VIII. Displayed output should be well mannered and well presented. Use appropriate comments and indentation in your source code.

Learning Objective

The AIM of this assignment is to have hands-on Pointers, dynamic memory allocation and multidimensional arrays and recursion.

Warning:

- If there is a syntax error in the code, zero marks will be awarded in that part of the assignment.
- Your code must be generic and handle all errors and exceptions.

Q.1. Find the Awkward Sum: (50 Marks)

You are tasked with working on a programming problem involving pointers and **DMAs** (Dynamic Memory Allocation). Your **objective** is to create a 2-D square grid of variable size, as specified by the user, and populate it with random values ranging from **1** to **99**. The dimensions of the grid should fall within the range of **5** to **9**.

Once the grid is created, your task is to identify combinations of consecutive prime numbers within the grid. A combination is defined as a sequence of at least **three** consecutive prime numbers, and it can exist **horizontally**, **vertically**, or **diagonally** within the grid.

Keep in mind that a number can be part of **multiple** combinations, and you need to consider **overlapping** elements. Your primary goals are to find the **sum** of all the remaining numbers in the grid that are **not** part of any combination, whether prime or non-prime. Additionally, you need to provide **statistical information**, such as the **size** of the **largest** combination and its **sum**, as well as determine which row or column contains the **highest number of combinations**.

1 - Prompt the user to input the desired size of the grid.

2 - Create a 2-D array with the specified dimensions provided by the user.

3 - Assign random values to each cell in the grid, ensuring they fall within the **range** of **1 to 99.**

4 - Identify combinations of consecutive prime numbers within the grid. (contains 30 Marks)

a. Check for combinations **horizontally** (in rows).

b. Check for combinations **vertically** (in columns).

c. Check for combinations **diagonally** (both directions).

5 - Ask the user which information they would like to retrieve:

a. If they choose the **largest combination**:

- Determine the **size** (number of consecutive prime numbers) of the largest combination found.

- Calculate the sum of the values in the **largest** combination.

b. If they choose the row or column with the most **combinations**:

- Determine which row or column contains the **highest number** of combinations.

c. If they choose the sum of all combinations and the remaining non-combination numbers:

- Calculate the **sum** of all the values in the **identified** combinations.(**Same Number Should not be added twice**)

- Calculate the **sum** of the remaining numbers that are **not** part of any combination

Provide the requested information based on the **user's choice**.

Note: Pay **attention** to the rules for determining **combinations**, and **ensure** that your code considers **horizontal**, **vertical**, and **diagonal sequences**.

Bonus Marks: (5 Bonus Marks)

We have a challenge for you. We will provide you with a text file that contains a series of random numbers arranged for a 7x7 grid. Your task is to read the numbers from the file and incorporate them into your program. In case of 7*7 grid, you need to update your menu to give the user the option to either generate random numbers or read the numbers from the file.

Q.2. Find the Determinant of a Rubik's Cube: (30 Marks)

Design a program that allows the user to create a **Rubik's Cube** of varying sizes, ranging from **2x2x2** to **5x5x5**. Each **Side** of Rubik's **Cube** would have a **2-D Square Matrix**. The program should utilize **pointers** for **cube creation**. After obtaining the desired size from the **user**, **initialize** all sides of the cube with **random values (1-99)**. Prompt the user to enter **values** for the **top side** of the cube. Next, calculate the **determinant** of each side and display the sum of **determinants** for each side, as well as the updated sum of all sides. You have to take the Determinant from the **last column** **(No marks would be awarded if the determinant is taken from any other column except the last one)**. Then the program should also tell that there was any **Singular Side** of the Cube, if so then **Display** those sides as well.

1. Ask the user for the **desired** size of the **Rubik's** Cube (2x2x2 to 5x5x5).
    - Prompt the user: "Please enter the size of the Rubik's Cube (2-5): "
    - Read and validate the **user's input**.

2. Create the Rubik's Cube dynamically based on the user's chosen size.
    - **Allocate memory in heap** for the Rubik's Cube structure.
    - Initialize the Rubik's Cube with random values (1-99).
    - Display the initial Rubik's Cube.
3. Prompt the user to enter **values** for the top side of the Rubik's Cube.
    - Ask the user to **input values** row by row.
    - Update the top side of the Rubik's Cube **accordingly**.
4. Calculate the **determinant** for each side of the Rubik's Cube using the **last column**. (contains 20 Marks)
    - Iterate through each side of the Rubik's Cube.
    - **Extract** the last column of each side.
    - Calculate the **determinant** of each side using the last column.
    - Display the side and its determinant.
5. Update and display the running sum of determinants.
    - **Display** the updated **sum** after each side's determinant.
6. Singular Check
    - Check if any side of Rubik's Cube was **Singular**, if so then Display it under **another** section displaying "**Singular Sides**".

Q.3. a) Create Your (~) operator and bitset manipulator: (12 Marks)

Question Restrictions: You are not allowed to use tilt (~) operator, bitset manipulator and Loops. You can only use functions and recursion for this task.

You must choose a random number in the range of 1 to 500 for this question. Then you have to convert it into a 16-bit binary format. You must repeatedly print it in binary format so that the first time it is printed, the leftmost bit is dropped, the remaining bits are then inverted and printed again, and so on until there is only one bit left. One example for output is given below for better understanding.

Input: 415

Output:

0000000110011111

111111001100000

00000110011111

1111001100000

000110011111

11001100000

0110011111

001100000

10011111

1100000

011111

00000

1111

000

11

0

b) Draw Me Out: (8 Marks)

Question Restrictions: You are not allowed to use Loops and setw(). You can only use functions and recursion for this task.

In this Question, you are required to print a pattern using recursion. The 9-row pattern is given to you as a reference. The character to be printed and number of size of pattern should be taken as Input from the user. The minimum size of is 3. Implement all necessary input validations on user input (i.e. user should not be allowed to enter not printable character like space).

Input: size=9, character='*'

Output:

```
    *
   **
  ***
 ****
*****
 ****
  ***
   **
    *
```

Q.4. (10 Marks)

In this problem, you have to sort an integer array using a recursive sorting algorithm (divide and conquer technique). Your program should define an array on run time (Dynamic memory allocation). After taking input from user, you have to apply your technique to sort it.