NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

Programming Fundamentals Spring- 2025

ASSIGNMENT - 1

"Greatness isn't given, it's built. Challenge yourself daily, learn relentlessly, and strive for more because settling is the enemy of progress"

Due Date: 19 Feb 2025
Time: 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 and Submission Guidelines: **Total Marks: 110**

1. Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss your problems with your colleagues (and instructors) on google classroom.
2. If there is a syntax error in the code, zero marks will be awarded in that part of the assignment.
3. Keep a backup of your work always that will be helpful in preventing any mishap and avoid last hour submissions
4. Displayed output should be well mannered and well presented. Use appropriate comments and indentation in your source code.
5. Please ensure that only concepts covered in class are used for the assignment. Topics such as functions, loops and arrays, which have not yet been studied, are not allowed. Additionally, specify all constraints at the end of each question.
6. 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.**
7. Combine all your work in one folder. The folder must contain only .cpp files (no binaries, no exe files etc.).
8. Rename the folder as ROLL-NUM_PROGRAM_SECTION (e.g., i230001_DS/AI/CS) and compress the folder as a zip file. (e.g., i230001_DS/AI/CS.zip). Strictly follow this naming convention, otherwise marks will be deducted.
9. Submit the .zip file on Google Classroom within the deadline.
10. The student is solely responsible for checking the final file for issues like corrupt

file, viruses in the file, or mistakenly exe sent. If we cannot download the file from Google Classroom, it will lead to zero marks in the assignment.

Honor Policy

Plagiarism is a grave academic offense that can severely undermine your academic integrity and reputation. Any instance of a student found to have plagiarized their assignment, whether from a peer or external source, will be subject to strict consequences. This may result in a zero score for the current or all assignments, or in severe cases, even a failure in the course. Furthermore, all instances of plagiarism will be promptly reported to the **Disciplinary Committee** for further action.

Note: Start early so that you can finish it on time.

Question : 01 **Marks : 20**

You were orphaned at an early age. To survive, you found work at a highly competitive **automated toy car factory**. Unlike simple workshops, this factory handles **multiple complex orders** each day with **strict priority rules** and **variable car models**.

Factory Rules:

There are **3 different car models**, each requiring a unique combination of parts:

1. **Standard Car (Model S):** 4 wheels, 1 car body, 2 figures
2. **Sports Car (Model SP):** 6 wheels, 1 car body, 1 figure (extra wheels for design)
3. **Family Car (Model F):** 4 wheels, 1 large car body (counts as 2 car bodies), 4 figures

Your Task:

Given the **available parts** and a **sequence of orders**, determine:

1. **Whether the orders can be fulfilled in the given priority.**
2. **If not, calculate the maximum number of each car model that can be built.**
3. **If even one car of any model cannot be built, state that production is not possible.**

Conditions:

1. **Order Priority:** Orders must be fulfilled in the sequence **Standard → Sports → Family.**
2. **Strict Resource Allocation:** You cannot use parts reserved for higher-priority orders in lower-priority ones.
3. **Partial Fulfillment:** If an order cannot be fully met, build as many cars as possible **before moving to the next model.**
4. **No Loops Allowed:** Only **if-else conditions** are permitted to solve the problem.
5. **Extra Constraint:** If **exactly 7 cars** of any model can be built, you must output a special message:
    - "Lucky production! Exactly 7 cars of Model [X] built."

Example:

Input:
Wheels = 60
Car Bodies = 15
Figures = 20
Orders: 7 Standard, 2 Sports, 3 Family
Output: Lucky production! Exactly 7 cars of Model S built.
Could not fulfill all orders. Built: 7 Standard, 2 Sports, 1 Family car.

Question : 02 **Marks : 20**

Imagine you are developing a highly secure authentication system for a classified government project where the password mechanism must adhere to strict security protocols. However, due to specific legacy system constraints, you are **NOT allowed to use**:

- **Functions**
- **Loops** (no `for`, `while`, `do-while`)
- **Switch statements**

You must rely solely on **`if` and `else` conditions** to implement the following password verification logic.

Password Requirements:

1. **Length Constraints:**
   - The password must be **at least 8 characters** and **no more than 16 characters** long.
   - The password **cannot start or end** with the same character.
2. **Character Composition:**
   - Must contain **at least one uppercase letter** (A-Z).
   - Must contain **at least one lowercase letter** (a-z).
   - Must contain **at least one digit** (0-9).
   - Must include **at least one special character** from the set: `@, #, $, %, &, *`.
3. **Structural Rules:**
   - The **first character** must be an **uppercase letter**.
   - The **last character** must be a **digit**.
   - Password should **not contain spaces**.
4. **Complexity Rules:**
   - Password **cannot contain three identical characters consecutively** (e.g., "aaa", "111", "%%%").
   - Password **must include at least one pair of consecutive characters** where the first is a **lowercase letter** and the second is an **uppercase letter** (e.g., "aB", "xY").

Program Requirements:

- Ask the user to input a password.
- Use **only `if` and `else` statements** to verify the password against all the criteria.
- Display **specific error messages** for each rule the password fails to meet.
- Display a success message if the password meets **all** the conditions.

Question : 03 **Marks : 10**

Determine whether a given 64-bit signed integer is even or odd under the following constraints:

1. **You cannot use:**
    - Bitwise operators (&, |, ^, ~, <<, >>)
    - Comparison operators (>, <, >=, <=, ==, !=)
    - Logical operators (&&, ||, !)
    - String conversion or manipulation
    - Built-in functions that directly determine parity (like is_even(), mod(), etc.)
    - **You must process the number in constant time (O(1) time complexity), regardless of its size.**
    - **The solution should work for both positive and negative integers.**
    - **You cannot precompute or hardcode any specific values (e.g., lookup tables).**

Question : 04 **Marks : 20**

You are given **four integer inputs**: A, B, C, and D. Based on these inputs, you need to determine the **final output** X using the following logic:

1. If the sum of A and B is **greater than** the product of C and D, calculate:
    - X = ((A^2 + B^2) % (C + 1)) - D
2. **Otherwise**, if C is **even** and D is **odd**, calculate:
    - X = (A * D + B) / (C + 1)
3. If none of the above conditions are true, and:
    - **(A is positive AND B is negative) OR (C is divisible by D)**, calculate:
        - X = (A + B - C) * (D % (A + 1))
4. If none of the conditions apply, set:
    - X = A + B + C + D

Finally, based on the value of X:

- If X is **greater than 1000**, output: "Result is OUT OF RANGE"
- If X is **exactly 0**, output: "ZERO POINT"
- Otherwise, output the exact value of X.

Question : 05 **Marks : 20**

Design and implement a text-based game called "The Ultimate Logic Maze," where the player navigates through a complex maze of decision points. The maze consists of multiple rooms connected in a non-linear fashion. Each room has specific conditions that determine the next possible room based on the player's choices.

Rules and Constraints:

1. **No loops, while, switch statements, functions, or arrays are allowed.**
2. Use only **if-else statements** to handle all the game logic.
3. The game must have at least **10 unique rooms**, each with different conditions.
4. Players make decisions based on options presented in each room (e.g., "Go left," "Pick the red key," "Open the hidden door").
5. The game should have multiple endings: **one winning path** and several losing paths.
6. Players can collect up to **3 virtual items** (e.g., key, map, token), tracked using variables (no arrays).
7. Each item must influence the game's logic at some decision point.

- Introduce a hidden room that can only be accessed if the player's collected items' variables satisfy a specific bitwise condition (e.g., (key & map) == token).
- Use bitwise operators such as AND (&), OR (|), XOR (^), and NOT (~) creatively.

Question : 06 **Marks : 20**

20 years ago, the **FAST Islamabad Campus** relied solely on **CNIC numbers** to uniquely identify students. However, with an increasing student population and evolving administrative needs, the management decided to enhance its data storage system.Now, apart from the CNIC, the system should extract **comprehensive demographic and regional details** automatically, based on the CNIC structure. This includes:

- **Personal Information:** Name, Gender, Date of Birth
- **Geographic Information:** Province, Division, District, Tehsil, Union Council
- **Family Information:** Family Tree Number, Household Type
- **Validation Mechanisms:** CNIC authenticity check, Birth Year validation, and error handling

Your Task:

Write a **C++ program** that:

1. **Inputs:**
    - Full Name of the student
    - CNIC Number (13 digits)
2. **Performs the Following Operations:**
    - **Validation:**
        1. Check if the CNIC contains exactly **13 digits** and no alphabets.
        2. Ensure the **birth year** (extracted from CNIC) is between **1980 and 2024**.
        3. Validate the **province code** (first two digits) with predefined codes.
    - **Extract Information:**
        1. **Gender:** Determined using the **last digit** (odd = Male, even = Female).
        2. **Date of Birth:**
            - Extracted from digits **3-8** (YYMMDD format).
            - Convert to a valid date (e.g., 990715 → 15th July 1999).
        3. **Province, Division, District, Tehsil, Union Council:**
            - Mapped based on the first **5 digits** of the CNIC.
            - If the combination is unrecognized, mark as "Unknown."
        4. **Family Tree:**
            - Middle **7 digits** represent the family number.
        5. **Household Type:**
            - If family number starts with 1 or 2 → **Joint Family**
            - If starts with 3 to 5 → **Nuclear Family**
            - Otherwise → **Extended Family**

Sample Output:

```
Enter Name of Student: Ayesha Khan
Enter CNIC of Student: 4210199907154

Name: Ayesha Khan
Gender: Female
Date of Birth: 15 July 1999
Province: Sindh
Division: Karachi Division
District: Karachi Central
Tehsil: Liaquatabad
Union Council: UC-7
Family Tree Number: 9990715
Household Type: Nuclear Family
```

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