NATIONAL UNIVERSITY
OF COMPUTER AND EMERGING SCIENCES
CHINIOT - FAISALABAD CAMPUS

CS-1002
Programming Fundamentals

Semester Fall 2024

FAST School of Computing

Note: Use of any sort of loops for the following questions is not permitted.

Question 1:

Sara wants to develop tools for her team at the tech company. Her team needs a tool that converts numbers from **binary (base 2)** or **octal (base 8)** into **decimal (base 10)**. However, the input numbers in binary or octal will always be **3, 4, or 5 digits long** to simplify their tasks, the system would handle all 3 cases itself and we don't want to burden the user by making them input the number of digits as well.

Your job is to help Sara by writing a C++ program that will prompt the user to input a **binary** or **octal** number (3, 4, or 5 digits long), and the program will convert that number into **decimal**.

Requirements:

- The user will first choose whether they are inputting a **binary (base 2)** or **octal (base 8)** number:
  - **Press 1** to enter a binary number.
  - **Press 2** to enter an octal number.
- Based on the user's selection, prompt them to input a number with **either 3, 4, or 5 digits**.
  - For **binary**, the number must only contain 0 and 1.
  - For **octal**, the number must only contain digits from 0 to 7.
- If the user inputs a number that isn't 3, 4, or 5 digits long it should display an error message and not product any other output.
- Use a **switch statement** to handle the user's selection of either binary or octal.
- Convert the valid input number to **decimal (base 10)** and display the result.
- If the user enters any choice other than 1 or 2 for the base selection, display an error message and terminate the program.

Question 2:

Monthly Income Tax Calculation for Pakistani Citizens

You are tasked with writing a program for the **Tax Advisory Portal** that helps Pakistani citizens calculate their monthly income tax liability. The tax calculation follows the Pakistani income tax system for both **salaried** and **non-salaried** individuals for the fiscal year 2023-2024. The program also considers a **25% tax rebate** for teachers employed by non-profit educational institutions.

Requirements:

- The user will input their **annual income** in Pakistani Rupees (PKR) and specify whether they are **salaried or non-salaried**.
- The user must also specify whether they are a **teacher working in a non-profit organization**
- Based on the input, the program will calculate the **annual income tax** using the tax brackets below.
- If the user is a teacher in a non-profit organization, the program will apply a **25% rebate** on the calculated annual tax.

- The program will output the **monthly tax deduction**, assuming the annual tax is divided equally over 12 months.
- If the annual income is less than PKR 600,000, the program should inform the user that no tax is deducted.

Tax Brackets for Salaried Individuals (2023-2024):

1. **Up to PKR 600,000**: No tax.
2. **PKR 600,001 to PKR 1,200,000**: 2.5% of the amount exceeding PKR 600,000.
3. **PKR 1,200,001 to PKR 2,400,000**: PKR 15,000 + 12.5% of the amount exceeding PKR 1,200,000.
4. **PKR 2,400,001 to PKR 3,600,000**: PKR 165,000 + 20% of the amount exceeding PKR 2,400,000.
5. **PKR 3,600,001 to PKR 6,000,000**: PKR 405,000 + 25% of the amount exceeding PKR 3,600,000.
6. **Above PKR 6,000,001**: PKR 1,005,000 + 32.5% of the amount exceeding PKR 6,000,000.

Tax Brackets for Non-Salaried Individuals (2023-2024):

1. **Up to PKR 600,000**: No tax.
2. **PKR 600,001 to PKR 1,200,000**: 5% of the amount exceeding PKR 600,000.
3. **PKR 1,200,001 to PKR 2,400,000**: PKR 30,000 + 12.5% of the amount exceeding PKR 1,200,000.
4. **PKR 2,400,001 to PKR 3,600,000**: PKR 180,000 + 17.5% of the amount exceeding PKR 2,400,000.
5. **PKR 3,600,001 to PKR 6,000,000**: PKR 390,000 + 22.5% of the amount exceeding PKR 3,600,000.
6. **Above PKR 6,000,001**: PKR 975,000 + 30% of the amount exceeding PKR 6,000,000.

Input:

- The program should prompt the user to enter their **annual income** in Pakistani Rupees (PKR).
- The user will specify whether they are **salaried** or **non-salaried**.
- The user will specify whether they are a **teacher in a non-profit organization** by entering 'yes' or 'no'.

Output:

- The program will display:
    1. The **annual income tax** based on the appropriate tax brackets (salaried or non-salaried).
    2. If the user is a teacher in a non-profit, apply the **25% tax rebate** on the calculated annual tax.
    3. The **monthly tax deduction** (annual tax after rebate divided by 12).

Example Scenario 1:

A **salaried employee** earns an annual salary of PKR 1,500,000 and is **not a teacher**. The program will calculate the tax based on the salaried brackets and display the monthly tax deduction.

Example Scenario 2:

A **non-salaried individual** earns PKR 2,000,000 annually and is a **teacher in a non-profit**. The program will calculate the tax using the non-salaried brackets, apply the 25% rebate, and then display the monthly tax deduction based on the reduced tax.

Constraints:

- The program must handle tax calculation for both salaried and non-salaried individuals.
- You **must use if-else** statements to determine tax brackets and rebates.
- **No loops** should be used in the solution.
- The program should perform input validation to ensure that the income entered is a positive number and correctly validate whether the user is salaried and/or a teacher in a non-profit.

Question 3:

FESCO Residential Electricity Bill Calculation

You are tasked with developing a program that calculates the **monthly electricity bill** for residential customers of FESCO (Federal Electric Supply Company) based on their **monthly consumption** in units. The calculation differs based on the number of units consumed, and it includes applicable taxes.

Requirements:

1. The user will input the following:
    - The **number of units consumed** in a month (should be a non-negative integer).
2. The bill calculation will follow these rules:
    - For consumption of:
        - **0 to 100 units**: PKR 5 per unit
        - **101 to 300 units**: PKR 7 per unit
        - **301 to 400 units**: PKR 24.40 per unit
        - **401 to 500 units**: PKR 24.91 per unit
        - **501 to 600 units**: PKR 26.15 per unit
        - **601 to 700 units**: PKR 27.59 per unit
3. Taxes:
    - The total bill will incur an **additional 10% tax** (for general tax).
4. The program should ensure the following:
    - Input validation to ensure that the number of units consumed is a non-negative integer.
5. The program should calculate the total bill based on the above rates, apply the taxes, and display the calculated amount to the user.

Example Inputs and Outputs:

- **Input:**
    - Units Consumed: **250**

Output:

- Total Bill (before tax): **PKR 1,700**
- Total Tax: **PKR 170**
- Total Bill (after tax): **PKR 1,870**

- **Input:**
  - Units Consumed: **350**

Output:

  - Total Bill (before tax): **PKR 8,720**
  - Total Tax: **PKR 872**
  - Total Bill (after tax): **PKR 9,592**

Notes:

- Make sure to handle the calculation logic using if-else statements to determine the applicable rates based on the number of units consumed and apply the tax calculations correctly.

Question 4:

Advanced Health Insurance Premium Calculator

Develop a program that calculates health insurance premiums based on the following criteria:

- **Age of the applicant**:
  - Under 18: 2,000 PKR per month
  - 18 - 30: 3,500 PKR per month
  - 31 - 50: 5,000 PKR per month
  - Above 50: 8,000 PKR per month
- **Pre-existing conditions**:
  - No pre-existing conditions: no additional charge
  - One pre-existing condition: 25% increase in premium
  - More than one pre-existing condition: 50% increase in premium
- **Smoker status**: An additional 20% if the applicant is a smoker.

The program should ask the user for their age, whether they have pre-existing conditions, and if they smoke. It should then calculate and display the total premium, including all adjustments. Ensure input validation is present for age and condition status.

Question 5:

Create a C++ program that assesses an individual's qualification for a personal loan based on the following criteria:

- **Annual income**: Must be at least 1,500,000 PKR.
- **Credit score**: Must be above 700.
- **Debt-to-income ratio**: Must not exceed 35%.
- **Employment status**: Must be employed for at least the last 2 years.

The program should ask for these inputs and use a series of nested if-else statements to determine:

- If the applicant is eligible for the loan.
- If not, provide a specific reason (e.g., income too low, credit score too low, high debt-to-income ratio, or insufficient employment duration).

Make sure that once there is enough information that deems the user unqualified for a loan the system must no longer take further inputs and stop and inform the user that they are ineligible.

Question 6:

Secure Login System with Multi-Layer Authentication

Design a program that simulates a multi-layer authentication system for a secure login. A user must input:

- A username
- A password
- A 2-factor authentication code (e.g., a 6-digit OTP)

The program should allow access if:

1. The password is correct.
2. When the username password matches, the 2-factor authentication code is entered correctly.
3. The user **does not** have any previous failed login attempts in the current session.

If any of these conditions fail, display an appropriate error message.

Hint: Use the NOT operator to check if there are **no** failed login attempts (!failedAttempts), and combine conditions for the password and OTP.

Username password and OTP all will be of int type.

Question 7:

Conditional Decryption Scheme

A secret agency has developed two different decryption schemes based on the case of the first character of a 5-character encrypted message. You are tasked with creating a program that decrypts this message.

- The user will input exactly **5 characters**.
- If the **first character** is a **capital letter (A-Z)**, the program will use **Scheme 1**.
- If the **first character** is a **lowercase letter (a-z)**, the program will use **Scheme 2**.

Scheme 1:

1. **Shift by ASCII value**: For each character in the string, subtract 3 from its ASCII value.
2. **Reverse the string**: Once all characters are shifted, reverse the entire string.
3. Print the decrypted message.

Scheme 2:

1. **Vowel replacement**: Replace all vowels in the string with the next vowel in the sequence:
    - a → e
    - e → i
    - i → o
    - o → u
    - u → a
2. **Swap case**: Convert all lowercase letters to uppercase and all uppercase letters to lowercase.

3. Print the decrypted message.

Input/Output Example:

Example 1:

- **Input**: "Bdfh7"
- The first character is a capital letter, so **Scheme 1** is used.
- **Decrypted message**: After applying Scheme 1, the output is "48?ec".

Example 2:

- **Input**: "aDc3F"
- The first character is lowercase, so **Scheme 2** is used.
- **Decrypted message**: After applying Scheme 2, the output is "EDC3f".

Additional Notes:

- You are required to ensure that the input has exactly 5 characters.
- Use ASCII values to handle character shifting.
- Only English alphabets and numeric digits (0-9) are considered valid input.

Question 8:

Smart Energy Billing System

A **smart energy billing system** is developed to optimize billing and load management based on the first digit of a user's **5-digit customer number**. The system analyzes the first digit and applies different scenarios to calculate the user's final bill.

- The user will input their **5-digit customer number** and monthly power usage in **kWh**.
- The program will determine the scenario based on the **first (leftmost) digit** of the customer number.

Load Management Scenarios:

1. **If the first digit is a prime number (2, 3, 5, 7)**:
    - The system will add a **10% discount** on the total bill if the user's usage is less than 300 kWh.
    - If the usage is above 300 kWh, the customer receives a **flat rate bill** with no discount.
2. **If the first digit is an odd number but not prime (1, 9)**:
    - The user will be charged **extra for peak-hour usage**.
    - The system will ask the user to input their **peak-hour consumption in kWh**.
    - If peak-hour consumption is more than 50% of the total monthly usage, a **15% surcharge** is applied on the total bill.
    - Otherwise, the user pays the normal rate with no surcharge.
3. **If the first digit is 0 or an even number (4, 6, 8)**:
    - The user is classified as a **low-risk consumer** and is **exempt from all surcharges**.
    - They will only be charged a flat rate for their monthly usage, with no discounts or extra charges.

Input/Output Example:

Example 1:

- **Input**: Customer Number: 35792, Monthly Usage: 280 kWh

- The first digit is 3, which is a prime number.
- Since the usage is less than 300 kWh, the user receives a **10% discount**.
- **Output**: "10% discount applied. Final bill: X"

Example 2:

- **Input**: Customer Number: 91234, Monthly Usage: 500 kWh
- The first digit is 9, which is an odd number but not prime.
- The system asks for peak-hour usage: 300 kWh.
- Since peak-hour consumption is 60% of the total, a **15% surcharge** is applied.
- **Output**: "15% surcharge applied. Final bill: Y"

Example 3:

- **Input**: Customer Number: 40856, Monthly Usage: 600 kWh
- The first digit is 4, which is an even number.
- The user is classified as low-risk and pays the flat rate with no adjustments.
- **Output**: "No discounts or surcharges. Final bill: Z"

You need to use to switch statements for scenarios.

Question 9:
Recursive Rightmost Digit Manipulation

You are tasked with creating a program that takes a **6-digit number** as input from the user and repeatedly modifies it based on the **rightmost digit** until only **one digit** remains. The rightmost digit is **discarded** after each step, and the process continues with the remaining number.

Process:

1. The user will input a **6-digit number**.
2. The program checks the **rightmost digit** of the number:
    - If the last digit is **even**, add 1 to the remaining number (after discarding the last digit).
    - If the last digit is **odd**, subtract 1 from the remaining number (after discarding the last digit).
    - If adding 1 results in the **remaining number gaining an extra digit**, subtract **2** instead.
3. Discard the rightmost digit and apply the same process to the new modified number.
4. Repeat this process until only **1 digit** remains.
5. The program will then print the final digit.

Input/Output Examples:

Example 1:

- **Input**: 999992
- The rightmost digit is 2 (even), so you add 1 to the remaining number (99999), getting 100000. Since adding 1 causes a digit overflow, you subtract 2 instead, resulting in **99997**.
- Now, the number is 99997. The rightmost digit is 7 (odd), so you subtract 1 from the remaining number (9999), getting **9998**.

- Now, the number is 9998. The rightmost digit is 8 (even), so you add 1 to the remaining number (999), getting **1000**. This adds a digit, so you subtract 2 instead, getting **998**.
- Now, the number is 998. The rightmost digit is 8 (even), so you add 1 to the remaining number (99), getting **100**. Since adding 1 adds a digit, subtract 2, getting **98**.
- Now, the number is 98. The rightmost digit is 8 (even), so you add 1 to the remaining number (9), getting **10**. This adds a digit, so you subtract 2 instead, getting **8**.
- The final result is **8**.

Example 2:

- **Input**: 123456
- The rightmost digit is 6 (even), so you add 1 to the remaining number (12345), getting **12346**.
- Now, the number is 12346. The rightmost digit is 6 (even), so you add 1 to the remaining number (1234), getting **1235**.
- Now, the number is 1235. The rightmost digit is 5 (odd), so you subtract 1 from the remaining number (123), getting **122**.
- Now, the number is 122. The rightmost digit is 2 (even), so you add 1 to the remaining number (12), getting **13**.
- Now, the number is 13. The rightmost digit is 3 (odd), so you subtract 1 from the remaining number (1), getting **0**.
- The final result is **0**.

Question 10:
Trip Planner for Weather Conditions

Design a trip planner for outdoor activities. The user inputs:

- Weather forecast: sunny, rainy, snowy, or foggy.
- Temperature in degrees Celsius.
- Wind speed in km/h.

The program should output whether the trip is allowed or not. The trip is **not allowed** if:

- The weather is **not** sunny or clear.
- The temperature is below 10°C or above 40°C.
- The wind speed exceeds 50 km/h.

If any of these conditions are met, the program should advise against the trip. If all conditions are met, display a message allowing the trip.

Hint: Use the NOT operator to negate the safe weather condition and check for extremes in temperature and wind speed.