CS201 Practice Exercises

Hello World! Program and Operators

Problem Statement # 01

Create a C++ program that begins by displaying the message "Hello, World!" on the screen. Subsequently, prompt the user to enter an integer, float number and character input. Upon receiving the input, display the entered integer, float and character on the monitor screen.

The program should encompass the following functionalities:

- **Display Hello word! Message:** The program should use the <iostream> header to incorporate standard input-output stream functionality. Upon execution, the program should display the hello Word! Message on the screens.
- **Input three Numbers:** Program should display the message to enter an integer input and then take an integer value as input from the user. Subsequently it should display the message to enter the float input and take input the float number and lastly, program diplays the message to take character data type and then take character input from the user.
- **Display the Message:** Once the program has taken the three data types it should displays the three data tapes i.e. integer, float and character on separate lines on the monitor screen.

Sample Data:

Integers: 1,2, 3 etc, Float: 2.65, 3.89 etc. Characters: a, b, c , A etc.

Console Screenshot (Reference):

```
Hello, World!
Please enter an integer: 5
Please enter a float: 3.14
Please enter a character: A
You entered the integer: 5
You entered the float: 3.14
You entered the character: A
```

Problem Statement # 02

Implement a C++ program that calculates the area and perimeter of different geometrical shapes. And displays the results as per the input values provided by the users.

The program should include the following functionalities:

- **Calculate Area and Perimeter of the Circle:** Upon execution, the program ask the user to input the radius of the circle. After getting the input it should calculate the circumference and the area of the circle. You can take the radius as float data type.

  Circumference of circle: $2\pi r$.

  Area of circle: πr^2

- **Area and the Perimeter of Rectangle:** Program should ask the user to take the length and width of the rectangle and calculate the area and perimeter of the Rectangle. You can take the length and width as float data type.

  Perimeter of Rectangle: 2 (Length + width).

  Area of Rectangle: Length x Width

- **Displays Result:** Once your program has taken input from the user it should calculate the result of area, perimeter and circumference of corresponding geometric shape and displays it on the screen.

Sample Data:

Float Length: 2.65, 3.89 etc.

Console Screenshot (Reference):

```
Circle Calculations
Enter the radius of the circle: 5
Circumference of the circle: 31.4159
Area of the circle: 78.5398

Rectangle Calculations
Enter the length of the rectangle: 10
Enter the width of the rectangle: 5
Perimeter of the rectangle: 30
Area of the rectangle: 50
```

Problem Statement # 03

Design a C++ program that enables users to convert temperature measurements between Fahrenheit, Celsius, and Kelvin. The program should ask the user to enter a temperature value in decimal data type in corresponding unit (Fahrenheit, Celsius, or Kelvin) and then convert it to the other two units.

The program should include the following three functionalities:

- **Input in Fahrenheit:** Upon execution, it should prompt the user to enter a temperature value in Fahrenheit. After receiving the input in doble data type then program should convert the temperature to Celsius and Kelvin by using the following formulas:

  Celsius = (Fahrenheit - 32) * 5/9

  Kelvin = Celsius + 273.15

  Finally, the program should display the converted temperatures in Fahrenheit and Kelvin to the user.

- **Input in Celsius:** Program should prompt the user to enter a temperature value in Celsius. After receiving the input in double data type then program should convert the temperature to Fahrenheit and Kelvin by using the following formulas:

  Fahrenheit = (Celsius * 9/5) + 32

  Kelvin = Celsius + 273.15

  Finally, the program should display the converted temperatures in Fahrenheit and Kelvin to the user.

- **Input in Kelvin:** it should prompt the user to enter a temperature value in Kelvin. After receiving the input in double data type, the program should convert the temperature to Fahrenheit and Celsius by using the following formulas:

  Fahrenheit = (Kelvin - 273.15) * 9/5 + 32

  Celsius = Kelvin - 273.15

  Finally, the program should display the converted temperatures in Fahrenheit and Celsius to the user.

Sample Data:

Float Temperature: 63.75, 63.5.89 etc.

Console Screenshot (Reference):

Your program should give the one of the following outputs.

Sample Output 1

```
Enter the unit of the temperature (use 'F', 'C', or 'K'): F
Enter temperature: 100
```

```
Temperature in Fahrenheit: 100.00°F
Temperature in Celsius: 37.78°C
Temperature in Kelvin: 310.93 K
```

Sample Output 2

```
Enter the unit of the temperature (use 'F', 'C', or 'K'): C
Enter temperature: 25
```

```
Temperature in Fahrenheit: 77.00°F
Temperature in Celsius: 25.00°C
Temperature in Kelvin: 298.15 K
```

Sample Output 3

```
Enter the unit of the temperature (use 'F', 'C', or 'K'): K
Enter temperature: 300
```

```
Temperature in Fahrenheit: 80.33°F
Temperature in Celsius: 26.85°C
Temperature in Kelvin: 300.00 K
```

Sample Output 4

```
Enter the unit of the temperature (use 'F', 'C', or 'K'): X
Enter temperature: 50
```

```
Invalid unit entered!
```

Problem Statement # 04

Develop a C++ program that calculates both simple and compound interest based on user-provided input for the principal amount, annual interest rate, time, and the number of times the interest is compounded per year. The program should prompt the user to enter these values and then display the calculated simple interest and compound interest. Ensure the program handles user inputs effectively and displays results clearly.

The program should include the following two functionalities:

- **Simple Interest:** Develop a C++ program that calculates simple interest based on user-provided input. The program should prompt the user to enter the principal amount, the annual interest rate (as a percentage), and the time (in years). Using the formula:

  $$\text{Simple Interest} = (\text{Principal} \times \text{Rate} \times \text{Time}) / 100$$

  the program will compute the simple interest and display the result. This functionality allows users to quickly determine the amount of interest they will earn or owe over a specified period, based on a fixed interest rate applied to the initial principal amount.

- **Compound Interest:** In addition to calculating simple interest, the program should also calculate compound interest based on user-provided input. The program will prompt the user to enter the principal amount, the annual interest rate (as a percentage) Rate , the time (in years), and the number of times the interest is compounded per year i.e. n. Using the formula:

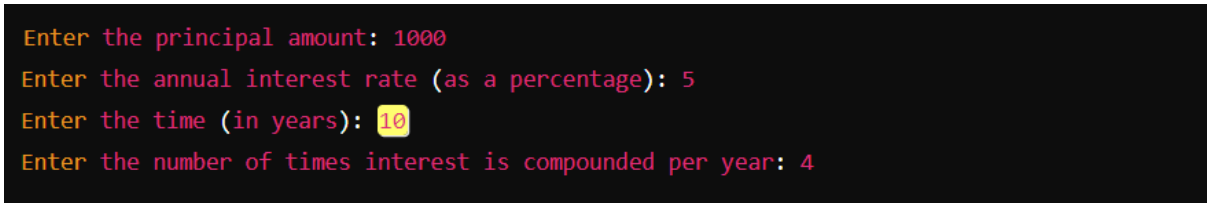
  $$\text{Compound Interest} = \text{Principal} \times (1 + \text{Rate} / n \times 100)^{n \times \text{Time}} - \text{Principal}.$$

  The program will compute the compound interest and display the result. This functionality helps users understand how their investment or loan will grow over time when interest is compounded periodically, providing a more accurate picture of financial growth compared to simple interest.

Sample Data:

Float Amount: 63.75, 63.5.89 etc.

Console Screenshot (Reference):



```
Simple Interest: $5000.00
Compound Interest: $6288.95
```

Problem Statement # 05

Develop a C++ program that calculates both parallel and series resistance for a given set of resistors. The program should prompt the user to input the number of resistors and their respective resistance values. It should then compute and display the equivalent resistance for both parallel and series configurations.

- The program should include the following two functionalities:
- **Series Resistance:** To calculate the series resistance, the program adds up all the resistance values provided by the user. The equivalent resistance in a series circuit is the sum of the individual resistances, making it straightforward to compute. The formula for the equivalent series resistance is:
    - $R_{series}=R_1+R_2+R_3+\ldots+R_n$
    - This method is useful in scenarios where resistors are connected end-to-end, and the current flows through each resistor sequentially. The total resistance increases with each additional resistor, reflecting the cumulative opposition to the current flow
- **Parallel Resistance:** For parallel resistance, the program calculates the reciprocal of the sum of the reciprocals of each resistor's resistance value. The equivalent parallel resistance is calculated using the formula:
    - $1/R_{parallel} = 1/R_1+ 1/R_2+ 1/R_3+\ldots+ 1/R_n$
    - This approach reflects the way parallel circuits distribute the current across multiple paths, reducing the overall resistance. The equivalent parallel resistance is always less than the smallest individual resistor in the network, highlighting the efficiency of parallel configurations in reducing total resistance. This calculation is essential in designing circuits where multiple resistors are connected across the same voltage source, ensuring an optimal and effective resistance value.

Sample Data:

Float Resistance: 43.75, 63.5.89 etc.

Console Screenshot (Reference):

```
Enter the number of resistors: 3
Enter the resistance values (in ohms):
Resistor 1: 10
Resistor 2: 20
Resistor 3: 30
Equivalent Series Resistance: 60.00 ohms
Equivalent Parallel Resistance: 5.45 ohms
```

If-else / Switch Statements

Problem Statement # 01

Write a C++ program to determine the eligibility of a person to vote based on their age. The program should ask the user to enter their age and then displays a message indicating whether they are eligible to vote or not. (Note: The person is eligible to vote if his/her age is greater than 18)

Problem Statement # 02

Write a C++ program that calculates the parking fee of a vehicle based on the duration of parking in a parking garage. The program should ask the user the number of hours the vehicle is parked and then calculate the parking fee. The fee structure is as follows:

- The first hour of parking is free.
- For every subsequent hour, a fee of Rs 200 is charged.

Problem Statement # 03

Write a C++ program to calculates the Body Mass Index (BMI) based on the user's weight (in kilograms) and height (in meters). Your program should ask the user to enter weight and height. (Note: The formula for calculating BMI is:

BMI = (Weight / Height * Height)

Display the user's BMI category according to the following criteria.

- Underweight (BMI < 18.5)
- Normal weight (BMI >= 18.5 and < 25)
- Overweight (BMI >= 25 and < 30)
- Obesity (BMI >= 30)

Problem Statement # 04

Write a C++ program for a coffee shop that allows customers to select their preferred beverage from a menu using a switch statement. The program should ask the user to enter a number corresponding to the coffee they want to order:

- 1 for Espresso
- 2 for Latte
- 3 for Cappuccino
- 4 for Mocha

Problem Statement # 05

Write a C++ program to represent a traffic light system using a switch statement. The program should ask the user to enter a number representing the colour of the traffic light: 1 for Red 2 for Yellow 3 for Green. Use a switch statement to display the corresponding message based on the colour:

- "Stop" for Red
- "Proceed with caution" for Yellow
- "Go" for Green

If the entered number does not match any colour, display an error message indicating an invalid input.

Loops

Problem Statement#1

Power Calculation:

Write a C++ program to calculate the power of a number x raised to an integer exponent n using a loop. The power of a number x^n is calculated by multiplying **x** by itself **n** times.

Functional Requirements:

- Prompt the user to enter the base number x and the exponent n.
- Calculate the result of x^n using a loop.
- Display the result of x^n.

Console Screenshot (Reference)

Your program output may look something like this:

```
Enter the base number: 2
Enter the exponent: 3
2 raised to the power of 3 is: 8
```

Problem Statement # 2

Prime Number Check:

Write a C++ program to check whether a given integer is a prime number or not. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Functional Requirements:

- Prompt the user to enter an integer.
- Check whether the entered number is a prime number.
- Display a message indicating whether the number is prime or not.

Console Screenshot (Reference)

Your program output may look something like this:

```
Enter a number: 4
4 is not a prime number.
```

```
Enter a number: 7
7 is a prime number.
```

Problem statement # 3

Password Verification:

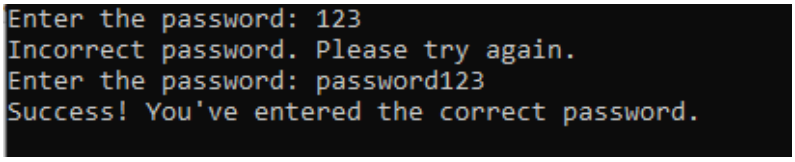
Write a C++ program that prompts the user to enter a password. The program should continue to prompt the user until they enter the correct password. Once the correct password is entered, the program should display a success message.

Functional Requirements:

- Define a correct password (e.g., "password123").
- Use a do-while loop to repeatedly prompt the user for a password.
- Compare the user input with the correct password.
- Display a success message when the correct password is entered.

Console Screenshot (Reference)

Your program output may look something like this:



Problem Statement#4

Shape Printing:

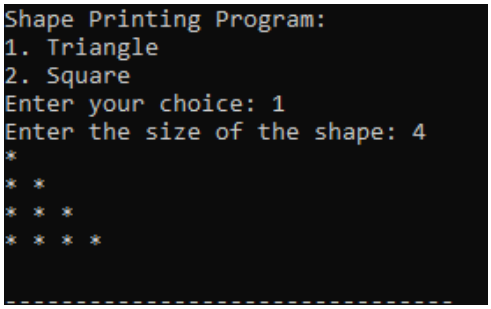
Write a C++ program to print a shape using nested loops. The shape to be printed can be a triangle, a square as shown in the console screenshot below. Your program should take the size or dimensions of the shape as input from the user.

Functional Requirements:

- Prompt the user to choose the shape to be printed (e.g., triangle, square).
- Depending on the selected shape, prompt the user to enter the size or dimensions of the shape.
- Use nested loops to print the shape.
- Display the printed shape.

Console Screenshot (Reference)

Your program output may look something like this:



Problem Statement#5

Grade Calculator:

Write a C++ program to calculate grades for students based on their scores. The program should continuously display a menu of options for the user to choose from, allowing them to calculate grades for multiple students. The grading criteria are as follows:

A score of 90 or above: A

A score of 80 to 89: B

A score of 70 to 79: C

A score of 60 to 69: D

A score below 60: F

Functional Requirements:

- Display a menu of options.
- Prompt the user to choose an option.
- Use a switch statement to perform the selected operation.
- If the user chooses to calculate grades:
- Prompt the user to enter the student's score.
- Determine and display the corresponding grade based on the grading criteria.
- Repeat the process until the user chooses to exit.

Console Screenshot (Reference)

Your program output may look something like this:

```
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice: 1
Enter student's score: 90
Grade: A
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice: 1
Enter student's score: 85
Grade: B
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice: 1
Enter student's score: 75
Grade: C
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice: 1
Enter student's score: 64
Grade: D
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice: 1
Enter student's score: 50
Grade: F
Grade Calculator Menu:
1. Calculate Grade
2. Exit
Enter your choice:
```

Functions

Problem Statement 1:

As a C++ developer working on a fitness tracking application, you're tasked with implementing a feature that calculates the average workout duration for a user's exercise sessions. To achieve this, you decide to write a function named calculateAverageWorkoutDuration().

The program will perform following functionalities.

- This function takes an array of integers representing the durations of individual exercise sessions and the size of the array as parameters.
- It then calculates and returns the average duration of the exercise sessions.
  Average = Sum of all durations / Total number of Sessions

For Example, array of exercise session durations
Int Array[]={40,50,60,20}

Problem Statement 2:

Develop a function to calculate the total cost of items in an online shopping cart.

- Functionality:
  1. Function named calculateTotalPrice().
  2. Takes an array of product prices and quantities as input.
  3. Calculates the total cost of items.

     For Example,

```
prices[] = {10.99, 5.49, 3.99}; // Example: Product prices
quantities[] = {2, 3, 1}; // Example: Quantities of each product
```

Problem Statement 3:

As an HR manager, create a function named calculateBonus() to evaluate employee performance and determine the bonus amount to be awarded. You have to prompt the following input parameters from user.

- **Input Parameters:**

1. employeeID: An integer representing the unique identifier of the employee.
2. performanceRating: An integer indicating the performance rating of the employee, typically on a scale from 1 to 5, with higher ratings indicating better performance.
3. salary: A double representing the employee's annual salary.

Bonus Calculation: The function calculates the bonus amount based on the following criteria:

- Employees with a performance rating of 4 or above are considered high performers and receive a bonus equivalent to 10% of their annual salary.
- Employees with a performance rating of 3 are considered to have average performance and receive a bonus equivalent to 5% of their annual salary.
- Employees with a performance rating below 3 are considered to have below-average performance and do not receive any bonus.

Problem Statement 4:

Problem Statement:

4. Develop a function to calculate interest earned for a savings account.

Functionality:

Function named calculateInterest().

The following parameters should be prompt from the user.

- Account balance (initial balance in the savings account).
- Interest rate (annual interest rate as a percentage).
- Time period (specified period for which interest is calculated).

Calculates interest earned over a specified period.

Problem Statement 5:

You're tasked with designing a function to calculate the total salary for employees.

Function Name: calculateSalary()

Input Parameters:

The following parameters should be prompt from user.

1. Hourly rate: The base rate of pay per hour for the employee.

2. Hours worked: The total number of hours worked by the employee during the pay period.
3. Overtime hours: The number of hours worked by the employee beyond the regular working hours, if applicable.
4. Overtime rate: The rate of pay per hour for the employee for extra hours.

Calculation:

5. The total salary is computed by considering both regular and overtime pay rates:

    1. Regular salary = Hourly rate * Hours worked
    2. Overtime salary = Overtime hours * Overtime Rate
    3. Total salary = Regular salary + Overtime salary

6. If there are no overtime hours, only regular salary is considered.

ARRAYS

Problem Statement 1.

You are required to write a C++ program to declare a One-dimensional integer Array. The program should prompt the user to provide the size of array and its elements accordingly during run-time. The output of the program should be to display the array elements in horizontal list, separated by a single tab.

Input:

- Size of array
- List of integers according to the size

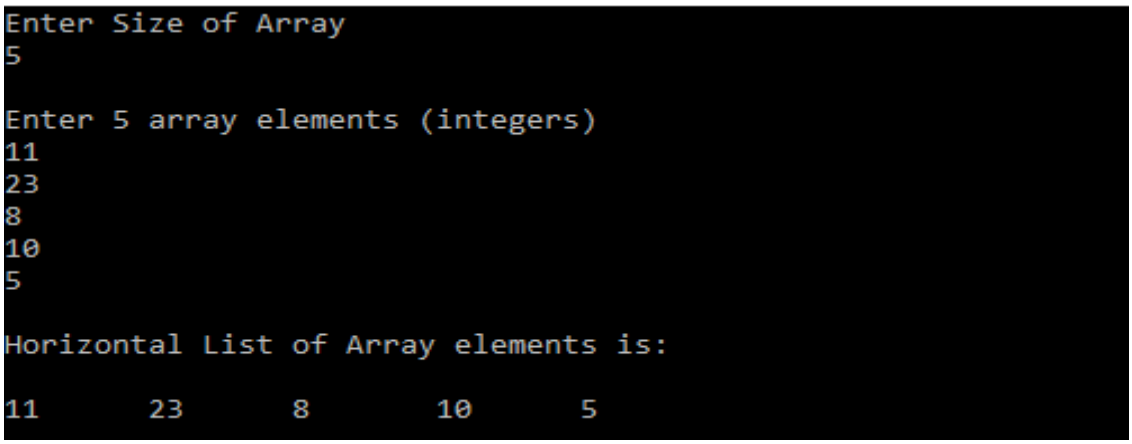
Output:

- A horizontal list of integers, separated by a single tab

Functional Requirements:

- Declaration of Integer array
- Prompt the user to enter the list of integer elements
- Read the contents of the list during run-time one by one using loop
- Display input list in horizontal fashion, separated by a single tab

Console Screenshot



Problem Statement 2

You are required to write a program in C++ to record the heights of 6 players in an array. Your program should find the maximum, minimum and average height of players by comparison and statistical formula. The output of the program should be to display the following data with prompting messages.

1. the list of player's heights
2. Maximum and minimum heights
3. Average height

Input:

- List of player's heights

Output:

- Maximum height of player
- Minimum height of player
- Average height of players

Functional Requirements:

- Declaration of floating-point array
- Enter the array elements by initializing the array
- Make comparison for finding the maximum and minimum height using the C++ conditional construct.
- Find the Average by using the statistical formula
- Display the list of heights with prompting message
- Display the Maximum height, minimum height and Average with prompting messages

Console Screenshot:

```
Height of players are:

5.7     5.4     5.6     5.5     5.8

Maximum height of player is 5.8

Minimum height of player is 5.4

Average height of player is 5.6
--------------------------------
```

Problem Statement 3:

You are required to write a program in C++ to input any unsorted integer array of size n. Your program should sort the given array in ascending order using the loop and conditional construct. The output of the program should be a sorted list.

Input:

- Size of array
- List of unsorted integers

Output:

- Sorted list of array elements with prompting message

Functional Requirements:

- Declaration of integer type array
- Enter the array elements during run-time
- Run a for-loop from 0 to size n-1
- Promote the smallest elements in the list by making comparisons
- At the end of loop, the array will get sorted in ascending order
- Display the sorted list with prompting message

Sample Data:

Enter the unsorted integer array elements:

11 8 15 5 20 3 17

The sorted integer list is:

3 5 8 11 15 17 20

Problem Statement 4.

You are required to write a program in C++ to declare a two-dimensional integer array, store elements in it during run-time using nested loops in row-major order. The output of the program should be to display the array elements in matrix form that is in row, column form.

Input:

- Two-dimensional list in row-major order

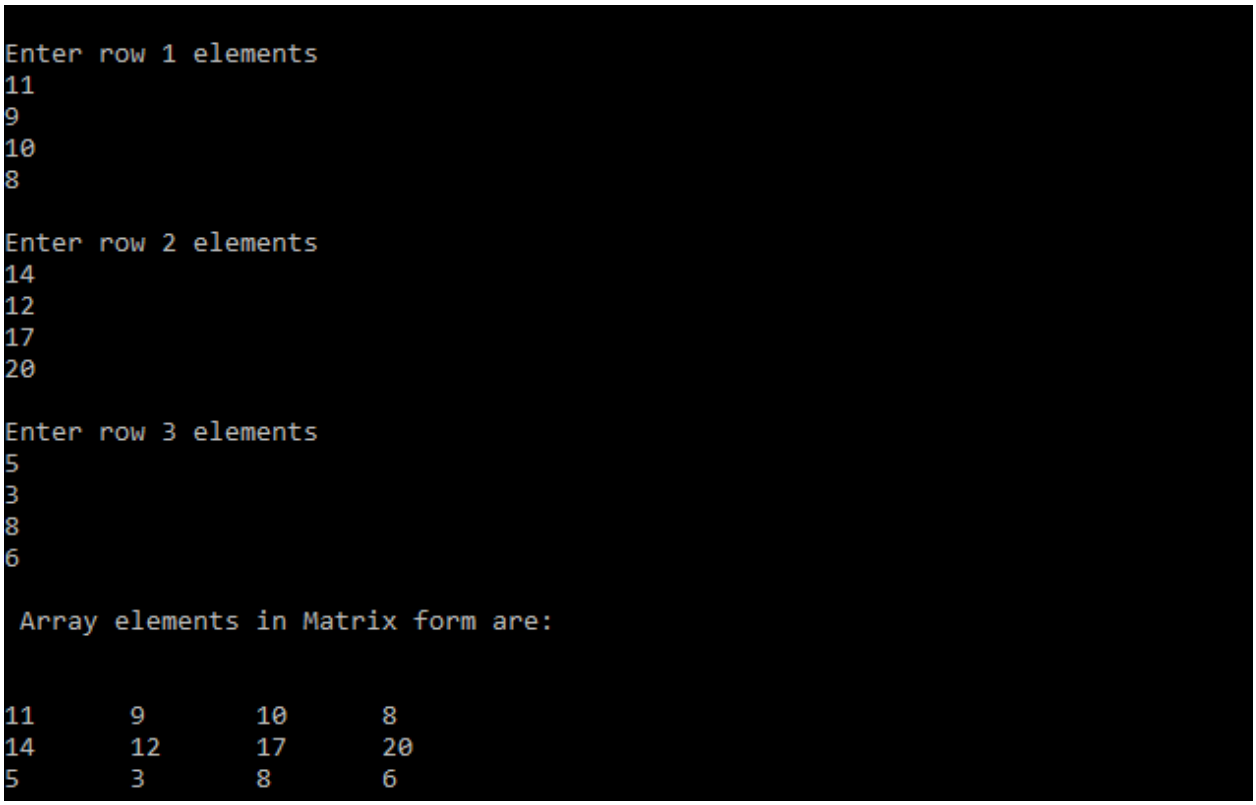
Output:

- Array elements arranged in matrix form

Functional Requirements:

- Declaration of 2-dimensional integer array
- Enter the array elements during run-time.
- Nested for-loops are used for entering elements such that the first row elements are entered first, then 2^{nd}, 3^{rd} and so on.
- Finally the array elements are displayed in a matrix-pattern using the nested for loops and escape sequence "\n" and "\t".

Console Screenshot:



Problem Statement 5.

You are required to write a program in C++ to record the score of **4 students** in **3 graded Quizzes** using 2-dimensional array during run-time. Find the sum and average score of each student and display the result in following row/column pattern.

	Quiz 1	Quiz 2	Quiz 3	Sum	Avg
Row 1	-	-	-	-	-
Row 2	-	-	-	-	-
!	-	-	-	-	-
!					

Input:

Enter score of 5 students in 3 graded quizzes using two-dimensional array

Output:

- Find the sum and average of each student

- Display the result in table form including the marks, sum and average of each student

Functional Requirements:

- Declaration of 2-dimensional floating array having 4 rows and 3 columns to keep the students Quizzes record
- Enter the Students score during run-time one by one in row-major order.
- Nested for-loops are used for entering the record such that the first student record entered first, then 2nd and so on.
- Finally the array elements are displayed in the given matrix-pattern using the nested for loops and escape sequence "\n" and "\t".
- During displaying the record of each student, the sum and average is also calculated and displayed in the position mentioned in patten.

Data Sample:

Enter Student 1 score:

6.5

7.5

9

Enter Student 2 score:

8.5

7.5

9

10

Enter Student 3 score:

5.5

9.2

7

6.5

Enter Student 4 score:

7.2

8.5

7.6

9.1

Students Quizzes Score with sum and Average:

	Quiz 1	Quiz 2	Quiz 3	Sum	Avg
Stud 1	6.5	7.5	9	23	7.6
Stud 2	-	-	-	-	-
!	-	-	-	-	-
Stud 4	-	-	-	-	-

POINTERS

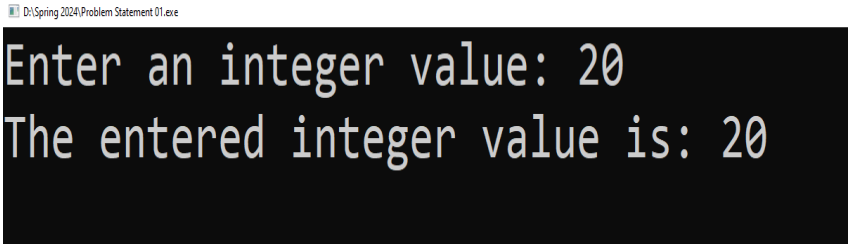
Problem Statement # 01

Create a C++ program that declares an integer variable and a pointer to that integer. The program should prompt the user to enter an integer value, store it in the integer variable, and then use the pointer to display the entered integer value on the screen.

The program should encompass the following functionalities:

- **Declare integer and pointer:** The program should declare an integer variable and a pointer to that integer.
- **Input integer value:** The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.
- **Display the value using pointer:** The program should assign the address of the integer variable to the pointer, and then use the pointer to display the entered integer value on the screen.

Sample Data:



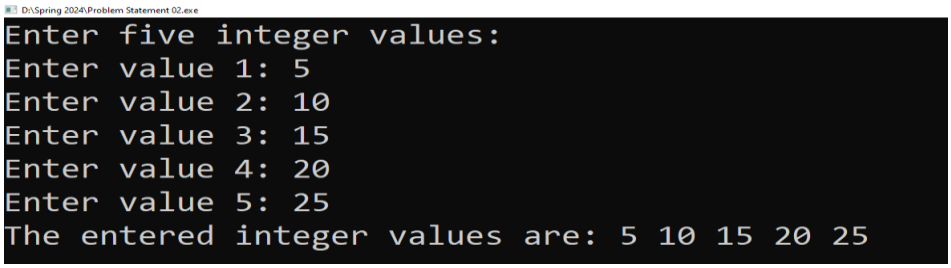
Problem Statement # 02

Create a C++ program that declares an array of integers and a pointer to an integer. The program should prompt the user to enter five integer values, store them in the array, and then use the pointer to display all the entered integer values on the screen.

The program should encompass the following functionalities:

- **Declare array and pointer:** The program should declare an array of integers with a size of 5 and a pointer to an integer.
- **Input array values:** The program should prompt the user to enter five integer values and store them in the array.
- **Display array values using pointer:** The program should assign the address of the array to the pointer, and then use the pointer to display all the entered integer values on the screen.

Sample Data:



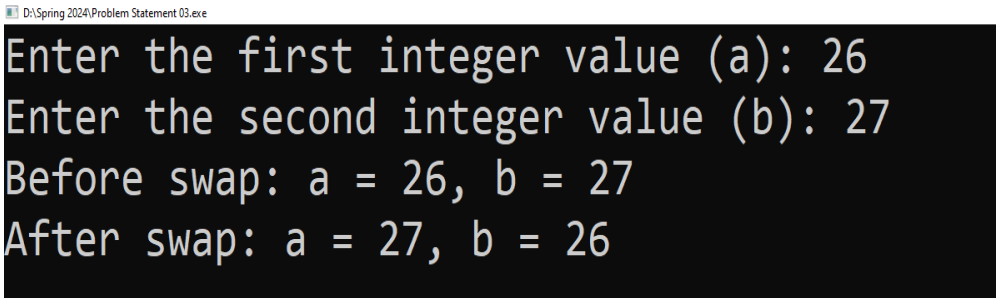
Problem Statement # 03

Create a C++ program that declares two integer variables and two pointers to integers. The program should prompt the user to enter two integer values, store them in the integer variables, and then use the pointers to swap and display the values.

The program should encompass the following functionalities:

- **Declare variables and pointers:** The program should declare two integer variables and two pointers to integers.
- **Input integer values:** The program should prompt the user to enter two integer values and store them in the integer variables.
- **Swap and display values using pointers:** The program should assign the addresses of the integer variables to the pointers, swap the values using the pointers, and then display the swapped values.

Sample Data:



Problem Statement # 04

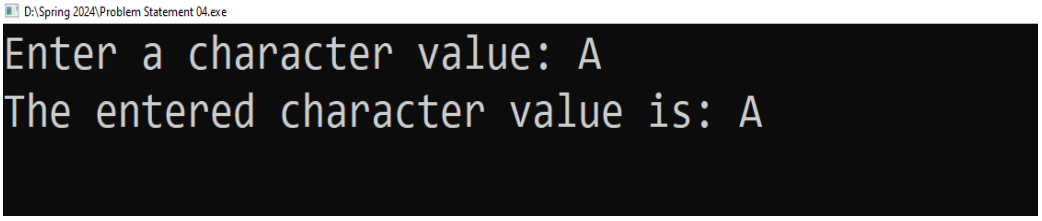
Create a C++ program that declares a character variable and a pointer to that character. The program should prompt the user to enter a character value, store it in the character variable, and then use the pointer to display the entered character value on the screen.

The program should encompass the following functionalities:

- **Declare character and pointer:** The program should declare a character variable and a pointer to that character.
- **Input character value:** The program should display a message prompting the user to enter a character value, and then take the character input from the user.

- **Display the value using pointer:** The program should assign the address of the character variable to the pointer, and then use the pointer to display the entered character value on the screen.

Sample Data:



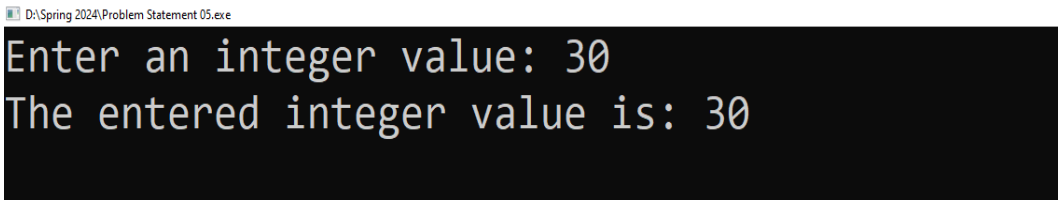
Problem Statement # 05

Create a C++ program that declares an integer variable and a pointer to that integer. The program should dynamically allocate memory for an integer, prompt the user to enter an integer value, store it in the allocated memory, and then display the value using the pointer. Finally, the program should free the allocated memory.

The program should encompass the following functionalities:

- **Declare pointer and allocate memory:** The program should declare a pointer to an integer and dynamically allocate memory for an integer.
- **Input integer value:** The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.
- **Display the value using pointer and free memory:** The program should store the entered value in the allocated memory, use the pointer to display the value, and then free the allocated memory.

Sample Data:



STRUCTS

Problem Statement # 01

Create a program using C++ language to store and display AddresBook information such as name, phone number, and email. The program should be capable to manage address book information of contacts like add, display, search and delete etc.

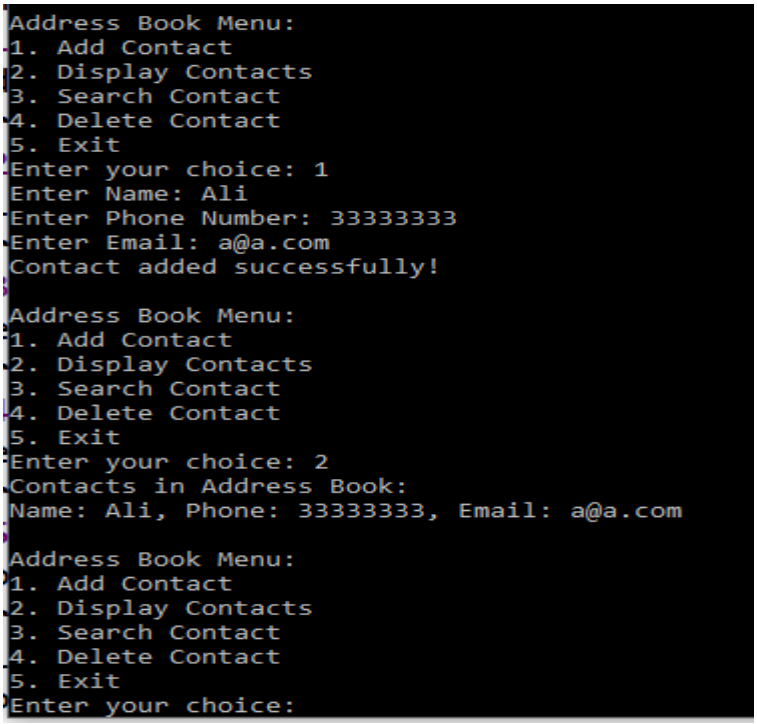
Structure Definition:

Define a structure **AddressBook** with the following fields:

- string name
- string phoneNumber
- string email

Functionality:

- **Add Contact:** Allow the user to add a new contact by entering name, phone number, and email.
- **Display Contacts:** Display all contacts in the address book.
- **Search Contact:** Search for a contact by name and display the contact details if found.
- **Delete Contact:** Delete a contact by name.
- **Exit:** Exit the program.
- **Menu-Driven Interface:**
- Present a menu to the user with options to add, display, search, delete, or exit.
- Loop until the user chooses to exit.
- **Sample Output:**



Problem Statement # 02

Create a program using C++ language for ABC banking system for customer bank accounts. The customer can create new account, deposit, with draw and check their account details. Use structure in C++ to implement this functionality according to the following requirements.

Structure Definition:

Define a structure **BankAccount**, which includes the following members:

- Account Number (int)
- Account Holder Name (string)
- Balance (double)

Functions:

- createAccount: Function to create a new account.
- deposit: Function to deposit money into an account.
- withdraw: Function to withdraw money from an account.
- displayAccount: Function to display the account details.
- **User Interface**:

The main function should provide a menu-driven interface to the user with options to:

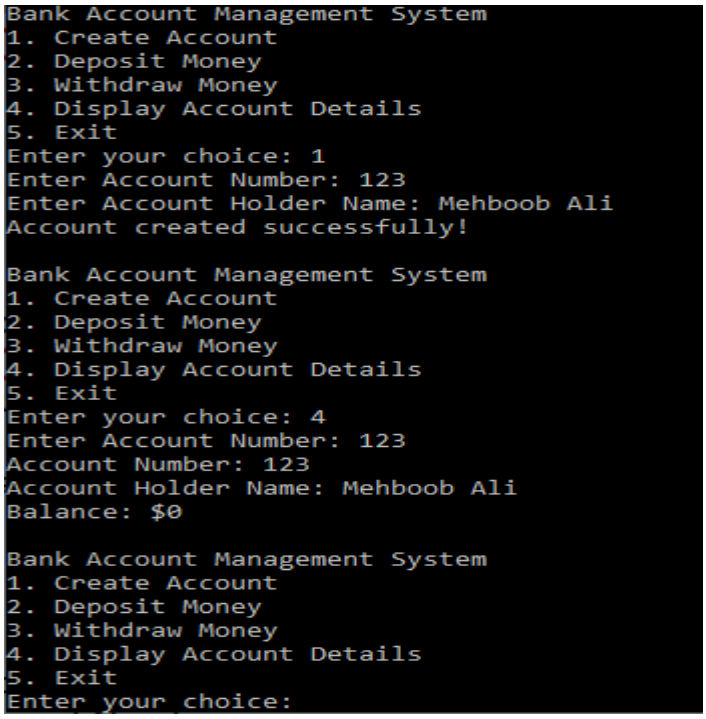
- Create a new account
- Deposit money

- Withdraw money
- Display account details
- Exit the program

Validation:

- Ensure that the account number is unique when creating a new account.
- Ensure that there is enough balance before allowing a withdrawal.

Sample Output:



Problem Statement # 03

Create a simple Product Inventory program in C++ involves a structure to represent a product and implementing functions to manage the inventory. Here's a detailed breakdown of the requirements and the implementation.

Structure Definition:

Define a structure **Product** with the following attributes:

- int id: A unique identifier for the product.
- string name: The name of the product.
- int quantity: The quantity of the product in stock.
- double price: The price of the product.

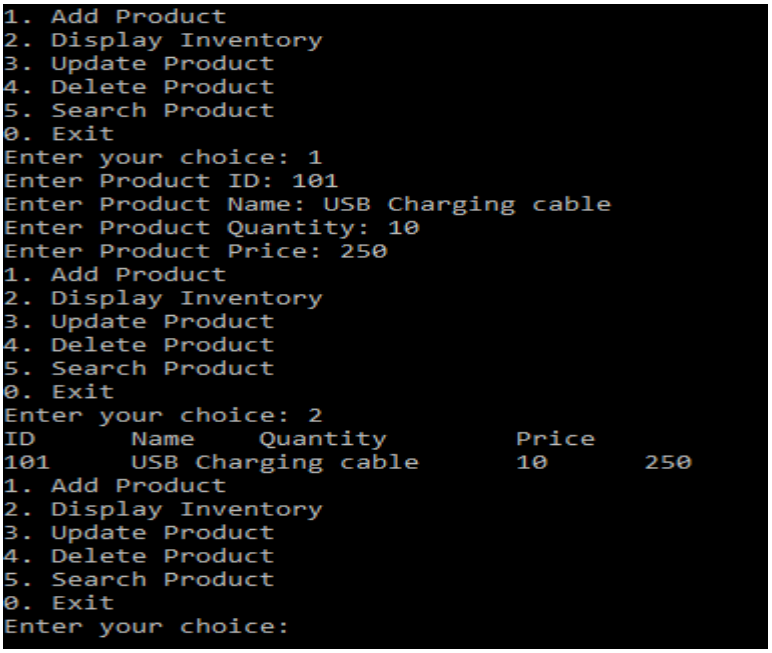
Functions:

- void addProduct(std::vector<Product>& inventory): Adds a new product to the inventory.
- void displayInventory(const std::vector<Product>& inventory): Displays all products in the inventory.
- void updateProduct(std::vector<Product>& inventory, int id): Updates the details of an existing product.
- void deleteProduct(std::vector<Product>& inventory, int id): Deletes a product from the inventory.
- Product* searchProduct(const std::vector<Product>& inventory, int id): Searches for a product by its ID.

Main Program:

- Implement a menu-driven interface to interact with the inventory system.
- Provide options for adding, displaying, updating, deleting, and searching for products.

Sample Output:



Problem Statement # 04

Create a simple Event calendar program in C++ which involves a structure to manage the event activities. Here's a detailed breakdown of the requirements and the implementation.

Structure Definition:

Define a structure **Event** with the following attributes:

- string event name: The name of the event.
- string date: The date of the event.
- string start time: The start time of the event.
- string end time: The end time of the event.
- string description: Description about the event.

Functions:

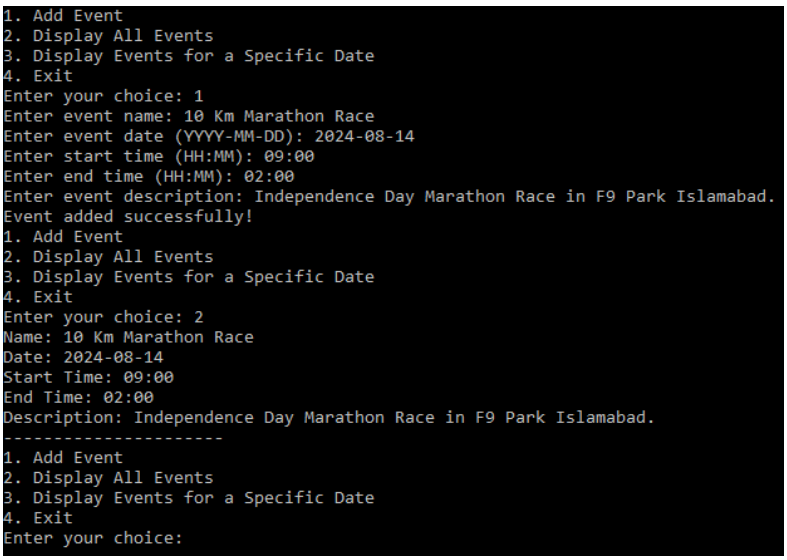
- Add an Event.
- Display all Events.
- Display Events for a Specific Date.
- **Input Validation**:

- Ensure that the date and time formats are valid.

- **Menu System:**

- Allow users to add events and view events.

Sample Output:



Problem Statement # 05

Create a Weather forecast program in C++ which involves a structure to predict about upcoming weather based on the data provided to the program. Here's a detailed breakdown of the requirements and the implementation.

Structure Definition:

Define a structure **Weather** with the following attributes:

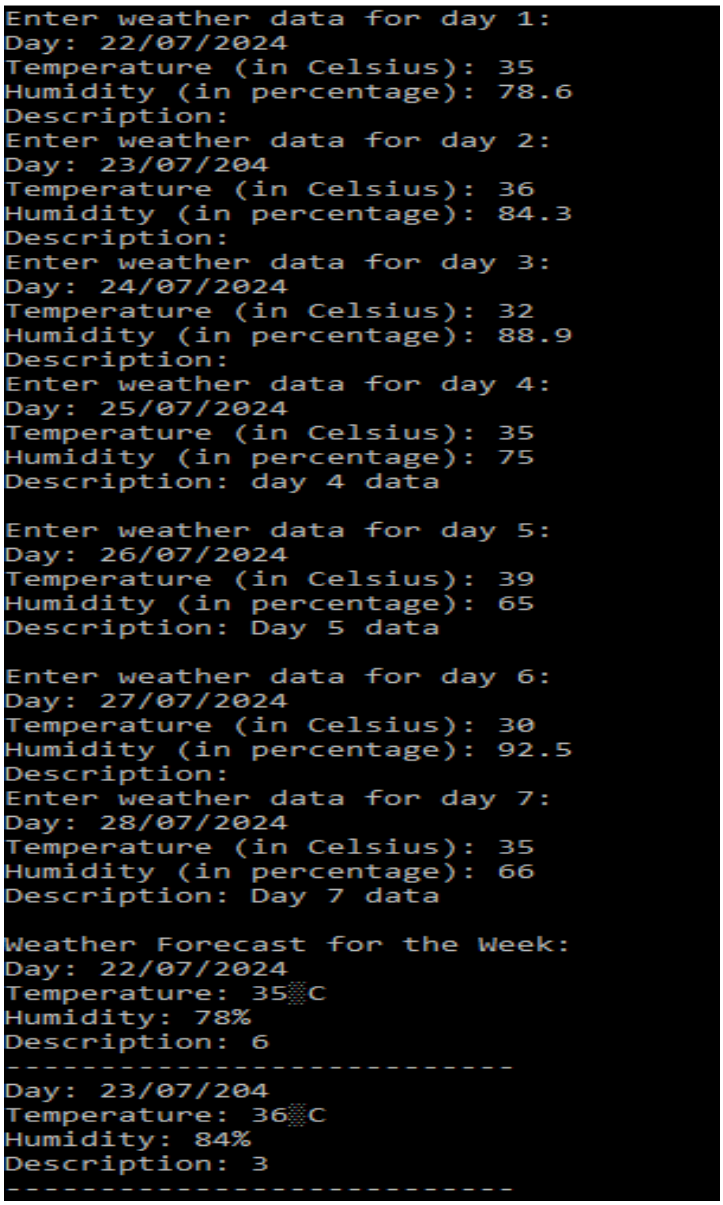
- string day: The weather for a particular day.
- float temperature: The temperature.
- Int humidity: The humidity of the weather.
- string description: Description about the weather.

Functions:

- The program should allow the user to input weather data (temperature, humidity, and description) for each day of the week.
- The program should display the weather forecast for the week after inputting the data.
- **Menu System:**

- Store weather data for a week.
- Allow the user to input weather data.
- Display the weather forecast for the week.

Sample Output:



File Handling

Problem Statement # 01

You are required to write a C++ program to merge the contents of two input files into a single output file. The program should prompt the user to provide the paths of the two input files and the desired path for the output file. After receiving the file paths, the program should read the contents of both input files line by line and write them sequentially into the output file.

Input:

- Path of the first input file.
- Path of the second input file.
- Path of the output file.

Output: A message indicating whether the merging was successful or if there were any errors encountered during the process.

Functional Requirements:

- Prompt the user to enter the path of the first input file.
- Prompt the user to enter the path of the second input file.
- Prompt the user to enter the path of the output file where the merged contents will be stored.
- Open the input files and the output file.
- Read the contents of the first input file line by line and write them to the output file.
- Read the contents of the second input file line by line and append them to the output file.
- Close all the files after merging is completed.
- Display a success message if the merging is done without any errors.

Error Handling:

- If any of the input files or the output file cannot be opened, display an error message indicating the issue.
- If the input file(s) are empty or do not contain any content, the program should still create the output file, but it will also be empty.

Console Screenshot (Reference):

```
Enter the path of first input file: input1.txt
Enter the path of second input file: input2.txt
Enter the path of output file: merged_output.txt
Files merged successfully.
```

Input File Contents:

input1.txt

```
This is the content of input file 1.
It contains multiple lines of text.
Each line will be merged into the output file.
```

Input2.txt

```
This is the content of input file 2.
It also contains multiple lines of text.
These lines will be appended to the output file after the content of input1.txt.
```

Output File Contents:

merged_output.txt

```
This is the content of input file 1.
It contains multiple lines of text.
Each line will be merged into the output file.
This is the content of input file 2.
It also contains multiple lines of text.
These lines will be appended to the output file after the content of input1.txt.
```

Problem Statement # 02

Design and implement a C++ program for user registration (signup) process using file handling. The program should allow users to create an account by providing a unique username and a password. The user information should be stored in a text file for future authentication.

Functionalities:

Registration Process:

1. The program prompts the user to enter a username and a password.
2. The username should be unique, i.e., not already registered by another user.
3. After successful registration, the program stores the user's information (username and password) in a text file named "users.txt".

Username Validation:

Before registering a new user, the program checks if the entered username is already taken by searching through the existing usernames in the "users.txt" file.

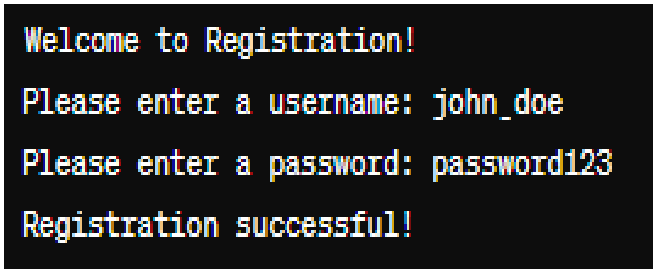
File Handling:

1. The program utilizes file handling to read from and write to the "users.txt" file.
2. Upon registration, the program appends the new user's information to the end of the file.

Error Handling:

The program should handle errors gracefully, such as file opening errors and invalid input.

Console Screenshot (Reference):



Problem Statement # 03

You are tasked with creating a C++ program that calculates the number of characters in a text file. The program should utilize file handling techniques to read the contents of the file and determine the total number of characters present.

The program should encompass the following functionalities:

- **File Opening:** Upon execution, the program attempts to open the specified text file containing the sample data. It verifies the existence and accessibility of the file before proceeding with the character count calculation.
- **Character Count Calculation:** After successfully opening the file, the program reads the contents character by character and counts the total number of characters present. It disregards any whitespace characters or special symbols and considers only alphanumeric characters.
- **File Closing:** Once the character count calculation is complete, the program closes the file to release any associated resources and ensure data integrity.

Sample Data:

Filename: sample_data.txt

Contents:

My student id is BC123456789. I study in Virtual University of Pakistan.

Console Screenshot (Reference):

```
Enter the name of the text file: sample_data.txt
Number of alphanumeric characters in the file: 36
```

Problem Statement # 04

You are tasked with developing a C++ program for managing a simple to-do list stored in a text file. The program enables users to append new tasks to the list, view existing tasks, and save modifications to the text file.

The program should include the following functionalities:

- **File Opening**: Upon execution, the program prompts the user to specify the name of the text file containing the to-do list. It then attempts to open the specified file, verifying its existence and accessibility.
- **Read from File**: If the file exists and can be opened successfully, the program reads and displays the current list of tasks to the user. Each task is displayed with a unique identifier and a brief description.
- **Write to File**: After displaying the existing tasks (if applicable), the program prompts the user to enter a new task to add to the list. Users input a brief description of the task, which is then appended to the end of the file as a new line.
- **File Closing**: Once the desired operations (reading and/or writing) have been completed, the program closes the file to release any associated resources and ensure data integrity.
- **Sample Data Format:** Define the format of tasks within the text file, such as a unique identifier (e.g., "Task 1:") followed by a brief description of the task. Each task should be stored as a new line in the file.

```
Task 1: Buy groceries
Task 2: Clean the house
Task 3: Finish report
```

Sample Output Screenshot (Reference):

```
Welcome to the To-Do List Manager!
Please enter the name of the to-do list file: todo.txt
File opened successfully.
Current Tasks:
Task 1: Buy groceries
Task 2: Finish report
Task 3: Call mom
Enter a new task description: Go for a run
Task added successfully.
To-Do List Manager exiting. File closed.
```

Problem Statement # 05

You are tasked with developing a C++ program that simulates a simple text editor. The program allows users to open a text file, perform read and write operations at different positions within the file, and display the current position of the file pointer.

The program should encompass the following functionalities:

- **File Opening**: Upon execution, the program prompts the user to specify the name of the text file they wish to open. It then attempts to open the specified file, verifying its existence and accessibility.
- **Read from File**: If the file exists and can be opened successfully, the program displays the current contents of the file to the user. It also provides an option for the user to navigate to a specific position within the file and display the character at that position.
- **Write to File**: After displaying the file contents (if applicable), the program prompts the user to enter new text data. It then allows the user to specify the position within the file where the data should be written. Upon successful writing, the program updates the file contents accordingly.
- **File Pointer Position**: The program provides an option for the user to query and display the current position of the file pointer (both get and put pointers). This functionality enables users to understand the current state of the file pointer during read and write operations.

Sample Output Screenshot (Reference):

```
Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 1
Enter file name: sample.txt
File opened successfully.

Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 2
File Contents:
Hello, this is a sample text file.
This file is for testing purposes.

Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 4
Current Get Pointer Position: 0
Current Put Pointer Position: 0

Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 3
Enter position to write: 6
Enter data to write: wonderful
Data written to file successfully.

Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 2
File Contents:
Hello, wonderful is a sample text file.
This file is for testing purposes.

Text Editor Menu:
1. Open File
2. Read from File
3. Write to File
4. File Pointer Position
5. Exit
Enter your choice: 5
Exiting program.
```

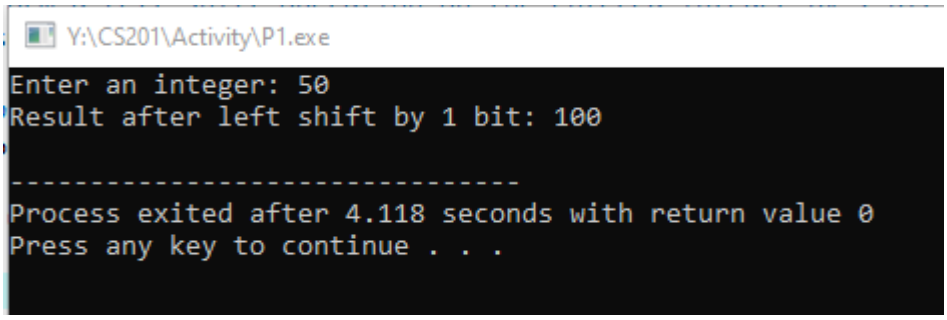
BIT MANIPULATION

Problem Statement # 01

Create a C++ program that takes an integer input from the user and performs a left shift operation on the integer by 1 bit. The program should then display the result. The program should encompass the following functionalities:

- Input integer value: The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.
- Perform left shift: The program should perform a left shift operation on the entered integer.
- Display the result: The program should display the result of the left shift operation.

Sample Data:

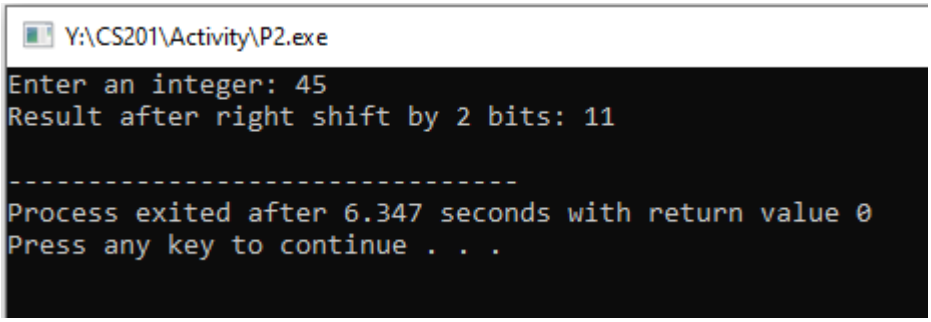


Problem Statement # 02

Create a C++ program that takes an integer input from the user and performs a right shift operation on the integer by 2 bits. The program should then display the result. The program should encompass the following functionalities:

- Input integer value: The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.
- Perform right shift: The program should perform a right shift operation on the entered integer.
- Display the result: The program should display the result of the right shift operation.

Sample Data:

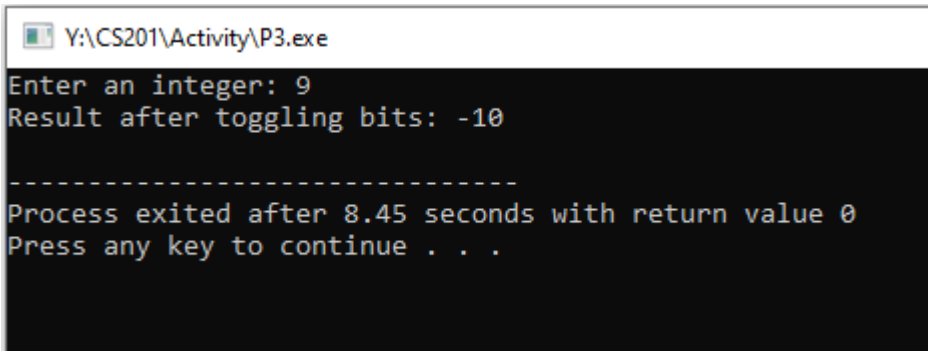


Problem Statement # 03

Create a C++ program that takes an integer input from the user and toggles (inverts) the bits of the integer. The program should then display the result. The program should encompass the following functionalities:

- Input integer value: The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.
- Toggle bits: The program should perform bitwise NOT operation on the entered integer.
- Display the result: The program should display the result of the bitwise NOT operation.

Sample Data:



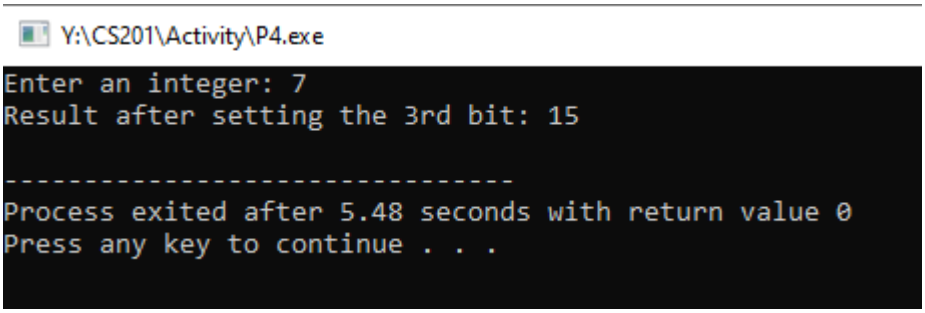
Problem Statement # 04

Create a C++ program that takes an integer input from the user and sets the 3rd bit of the integer (from the right, 0-based index). The program should then display the result. The program should encompass the following functionalities:

- Input integer value: The program should display a message prompting the user to enter an integer value, and then take the integer input from the user.

- Set the 3rd bit: The program should perform a bitwise OR operation to set the 3rd bit of the entered integer.
- Display the result: The program should display the result after setting the 3rd bit.

Sample Data:

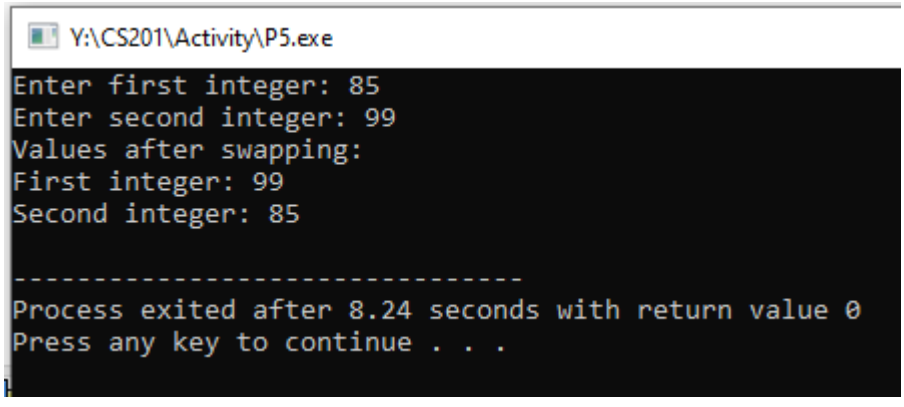


Problem Statement #05

Create a C++ program that takes two integer inputs from the user and swaps their values using bitwise XOR operation without using a temporary variable. The program should then display the swapped values. The program should encompass the following functionalities:

- Input integer values: The program should display messages prompting the user to enter two integer values, and then take the integer inputs from the user.
- Swap using XOR: The program should use the bitwise XOR operation to swap the values of the two integers.
- Display the result: The program should display the swapped values of the integers.

Sample Data:



CLASSES

Problem Statement # 01

Create a program using C++ language to store and display student information such as name, roll number, and marks in three different subjects. You need to create a **student** class, its data members and member functions. All data members must be private.

Class Name: Student

Data Members:

- **name** (string): stores the name of the student
- **rollNumber** (int): stores the roll number of the student
- **marks1**, **marks2**, **marks3** (float): store the marks of the student in three subjects

Methods:

- Setter functions to take input for student details (name, roll number, marks in three subjects)
- Getter functions to retrieve student information.
- **calculateTotalMarks():** Calculates the total marks obtained by the student
- **displayInfo():** Displays the student's information including name, roll number, marks in each subject, and total marks

Sample Output:

```
Enter name: Ali
Enter roll number: 123
Enter marks in Subject 1: 55
Enter marks in Subject 2: 67
Enter marks in Subject 3: 84

Student Information
Name: Ali
Roll Number: 123
Marks in Subject 1: 55
Marks in Subject 2: 67
Marks in Subject 3: 84
Total Marks: 206
```

Problem Statement # 02

Develop a program in C++ language to manage bank accounts with features like deposit, withdrawal, and display balance. You need to create a **BankAccount** class, its data members and member functions. All data members must be private.

Class Name: BankAccount

Data Members:

- **accountNumber (int):** stores the account number of the bank account
- **balance (float):** stores the current balance of the bank account

Methods:

- setter functions to store account number and balance.
- Getter functions to retrieve account number and balance.
- **deposit(float amount)**: Adds the specified amount to the account balance
- **withdraw(float amount)**: Subtracts the specified amount from the account balance if sufficient balance is available; otherwise, displays an error message

Sample Output:

```
Account Number: 12345
Balance: 1500
Account Number: 12345
Balance: 1300
Insufficient balance.
Account Number: 12345
Balance: 1300
```

Problem Statement # 03

Create a program using C++ language to perform geometric calculations on rectangles, such as calculating area and perimeter. You need to create a **Rectangle** class, its data members and member functions. All data members must be private.

Class Name: Rectangle

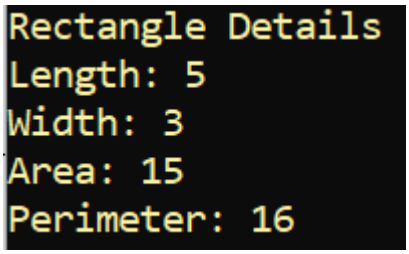
Data Members:

- **length**, **width** (float): store the dimensions of the rectangle

Methods:

- Getter/Setter functions to take input and retrieve data of length and width data members.
- **calculateArea()**: Calculates the area of the rectangle using length and width
- **calculatePerimeter()**: Calculates the perimeter of the rectangle using length and width
- **displayDetails()**: Displays the details of the rectangle including length, width, area, and perimeter

Sample Output:

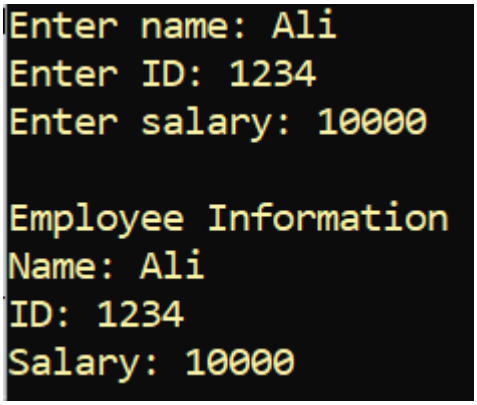


Problem Statement # 04

Develop a program using C++ to manage employee information, including name, ID, and salary.

- **Class Name:** Employee
- **Data Members:**
    - **name** (string): stores the name of the employee
    - **id** (int): stores the employee's ID number
    - **salary** (float): stores the employee's salary
- **Methods:**
    - Getter/Setter functions to take input and retrieve data of employee
    - **inputDetails()**: Takes input for employee details (name, ID, salary)
    - **displayInfo()**: Displays the employee's information including name, ID, and salary

Sample Output:



Problem Statement # 05

Create a program to convert temperatures between Celsius and Fahrenheit.

Class Name: TemperatureConverter

Data Members:

- **celsius**, **fahrenheit** (float): store temperatures in Celsius and Fahrenheit

Methods:

- Getter/Setter functions to take input and retrieve data of Celsius and Fahrenheit data members.
- **celsiusToFahrenheit(float celsius)**: Converts Celsius temperature to Fahrenheit
- **fahrenheitToCelsius(float fahrenheit)**: Converts Fahrenheit temperature to Celsius
- **displayConversion(float celsius, float fahrenheit)**: Displays converted temperatures from Celsius to Fahrenheit and vice versa

Sample Output:

```
Celsius: 30 -> Fahrenheit: 86
Fahrenheit: 86 -> Celsius: 30
```