Worksheet 1

24030183

**A logo with blue and red text

AI-generated content may be incorrect.A red sign with white text

AI-generated content may be incorrect.**

SUBMITTED BY: Sujal Adhikari  
STUDENT ID: 24030183  
SUBMITTED TO:   
SUBMITTION DATE:

# **Introduction**

This document provides a detailed explanation of **Coursework-1: C++ Basics** for the module **Programming in C++ (UFCFGL-30-1)**. The coursework is focused on fundamental concepts of C++ programming, including data types, control structures, and arrays. The tasks included in this coursework are designed to test the understanding and implementation of these fundamental programming concepts.

The coursework consists of three major programming exercises:

* **Task 1:** Exercises on data types and conditional statements.
* **Task 2:** Exercises on control statements, including a program to check for bouncy numbers.
* **Task 3:** Exercises on arrays, including a cinema ticket booking system.

Each task requires applying key C++ concepts, handling user input, and ensuring proper validation. The document will include problem statements, implemented solutions, and their corresponding outputs to demonstrate the correctness and efficiency of the solutions.

# **QUESTION 1.1**

1. Write a program that takes a temperature value from the user. It should then allow the user to choose between Celsius (C) and Fahrenheit (F) for conversion. After the user selection, it should then convert the entered temperature to the chosen scale and display the result.

Use appropriate data types for temperature and handle error like non-numeric input.

Use the following formula for conversion:

F = (C x 9/5) + 32

C = (F - 32) x 5/9

# **Code Implementation**

A screenshot of a computer program

AI-generated content may be incorrect.

# **Code Explanation and Logic**

This program is designed to take a temperature input from the user and convert it to either Celsius or Fahrenheit based on the user's choice. Below are the key concepts and technical terms used in the code:

1. **Input Handling** - The program uses cin to take numerical input for the temperature and a character input for the conversion choice.
2. **Data Types** - The temperature is stored in a double data type to allow decimal precision.
3. **Conditional Statements (if-else)** - The program uses an if-else if structure to check the user's choice ('C' or 'F') and perform the appropriate conversion.
4. **Mathematical Operations** - The formula used for temperature conversion:
   * Fahrenheit to Celsius: C = (F - 32) \* 5 / 9
   * Celsius to Fahrenheit: F = (C \* 9 / 5) + 32
5. **Error Handling** - If the user enters an invalid choice, an error message is displayed.

# Output

A screenshot of a computer

AI-generated content may be incorrect.

# **QUESTION 1.2**

1. Write a C++ program to implement a number guessing game with different difficulty levels.

Easy difficulty ranges from 1-8, medium from 1-30, hard from 1-50.Then,generate a random number to check if the guess is correct based on the user's selection.

# **Code Implementation**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

# **Code Explanation and Logic**

This program implements a **number guessing game** where the user selects a difficulty level, and the program generates a random number within the specified range for that level.

**Key Concepts and Technical Terms Used**

1. **Random Number Generation (rand())**
   * srand(time(0)) initializes the random number generator with the current time to ensure different results on each run.
   * rand() % range + 1 is used to generate numbers within different difficulty levels:
     + Easy: rand() % 8 + 1 (Range: 1-8)
     + Medium: rand() % 30 + 1 (Range: 1-30)
     + Hard: rand() % 50 + 1 (Range: 1-50)
2. **User Input Handling (cin)**
   * The program asks the user to enter their difficulty level (e, m, h).
   * Then, it takes a numerical guess as input.
3. **Switch-Case Statement (switch)**
   * Used to handle different difficulty levels efficiently.
   * Each case executes the corresponding logic based on the user's difficulty selection.
4. **Conditional Statements (if-else)**
   * After selecting a difficulty level, the program checks if the user's guess matches the randomly generated number.
   * If the guess is correct, it prints a success message.
   * If incorrect, it reveals the correct number.
5. **Error Handling (default in switch)**
   * If the user enters an invalid choice, the program displays an error message and asks them to restart.

# Output

**A screenshot of a computer

AI-generated content may be incorrect.**

# **QUESTION 1.3**

1. Write a program that reads an array of integer numbers from the user and sorts the numbers in the ascending order.

# **Code Implementation**

A screenshot of a computer

AI-generated content may be incorrect.

# **Code Explanation and Logic**

**This program takes an array of integers from the user, sorts them in ascending order, and then displays the sorted array.**

**Key Concepts and Technical Terms Used**

1. **Array Declaration (int arr[100])**
   * **A fixed-size array of 100 elements is declared to store user input.**
2. **User Input Handling (cin)**
   * **The user is asked how many numbers they want to enter.**
   * **A validation check ensures the number is between 1 and 100 to prevent errors.**
3. **Looping (for loop)**
   * **A for loop is used to take input from the user and store the values in the array.**
   * **Another for loop is used to display the sorted output.**
4. **Sorting (sort() function from <algorithm>`)**
   * **The sort(arr, arr + n); function is used to sort the array in ascending order.**
   * **This is a built-in function from the <algorithm> header in C++.**
5. **Error Handling (if condition)**
   * **If the user enters an invalid number of elements (n <= 0 or n > 100), the program displays an error message and exits with return 1.**

# Output

**A screenshot of a computer

AI-generated content may be incorrect.**

# **QUESTION 1.4**

1. Write a program that reads a number from the user and based on the user input, it says what day of the week it is, Sundays being 1 and Saturdays being 7. You system should give appropriate response for invalid input entries.

# **Code Implementation**

A screenshot of a computer

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

# **Code Explanation and Logic**

**This program reads an integer from the user and outputs the corresponding day of the week based on the user's input. If the input is outside the valid range (1-7), it displays an appropriate error message.**

**Key Concepts and Technical Terms Used**

1. **Input Handling (cin)**
   * **The program prompts the user to enter a number between 1 and 7, representing the days of the week.**
   * **The input is stored in the variable day.**
2. **Switch-Case Statement**
   * **The switch-case statement is used to map the user's input to the corresponding day of the week:**
     + **case 1: corresponds to Sunday**
     + **case 2: corresponds to Monday, and so on until case 7: for Saturday.**
   * **If the user enters a valid number (1-7), the corresponding day is printed.**
3. **Error Handling (default case)**
   * **The default case is used to handle invalid inputs:**
     + **If the user enters a number that is not between 1 and 7, the program outputs "Invalid day of the week! Please enter a number between 1 and 7."**
4. **Break Statement**
   * **The break statement is used after each case to prevent fall-through, ensuring only the corresponding day's name is printed.**

# Output

**A screenshot of a computer

AI-generated content may be incorrect.**

# **QUESTION 2.1**

1. Create a program that takes a positive integer as input and determines whether it's a "bouncy number". A bouncy number is one where the digits neither consistently increase nor consistently decrease when read from left to right. For example:

* 123 is NOT bouncy (digits consistently increase)
* 321 is NOT bouncy (digits consistently decrease)
* 120 is bouncy (neither consistently increasing nor decreasing)

# **Code Implementation**

A screenshot of a computer code

AI-generated content may be incorrect.

# **Code Explanation and Logic**

This program determines whether a positive integer is a **bouncy number**. A **bouncy number** is one where the digits neither consistently increase nor consistently decrease when read from left to right.

**Key Concepts and Technical Terms Used**

1. **Function Definition (isBouncy)**
   * The isBouncy function takes an integer num as input and determines if it's a bouncy number.
   * **Input Handling (cin)**: The program reads a positive integer from the user and stores it in the variable num.
2. **Early Return for Numbers Below 100**
   * If the number is less than **100**, it returns false immediately. This is because bouncy numbers must have at least three digits to compare the increasing and decreasing trends.
3. **Digit Extraction and Comparison**
   * The program extracts the digits of the number starting from the last digit. This is done by:
     + Using the modulo operation (num % 10) to get the last digit.
     + Dividing the number by 10 (num /= 10) to remove the last digit.
4. **Tracking Trends (Increasing and Decreasing)**
   * Two boolean variables, increasing and decreasing, are used to track whether the digits are increasing or decreasing:
     + If a digit is **greater** than the previous digit, it sets decreasing = true.
     + If a digit is **less** than the previous digit, it sets increasing = true.
5. **Early Exit if Both Trends are Found**
   * If both trends (increasing and decreasing) are found while checking the digits, the program immediately returns true because it confirms the number is bouncy.
6. **Final Check**
   * After the loop finishes, if neither trend is found (i.e., digits are consistently increasing or decreasing), the function returns false.
7. **Output Handling (cout)**
   * In the main function, based on the result from the isBouncy function, the program outputs whether the number is a **bouncy number** or not.

**Example**

For the number **120**:

* **Digit comparisons**: 1 (less than 2), 2 (greater than 0) → increasing and decreasing trends detected → it's bouncy.

For the number **123**:

* **Digit comparisons**: 1 (less than 2), 2 (less than 3) → digits are increasing → not bouncy.

**Code Output Example**

* For input 120, the output will be:  
  120 is a bouncy number.
* For input 123, the output will be:  
  123 is NOT a bouncy number.

# Output

A screenshot of a computer

AI-generated content may be incorrect.

# **QUESTION 3.1**

1. Write a program that manages a cinema ticket booking system. The program should display a 5x5 seating arrangement where:
   1. Available seats are marked with 'O'
   2. Booked seats are marked with 'X'

Program should:

* 1. Display the current seating arrangement
  2. Ask user for row and column number (1-5) for booking
  3. Mark that seat as booked ('X')
  4. Show updated seating after each booking
  5. Display error if user selects already booked seat
  6. Display error if user enters invalid row/column numbers

# **Code Implementation**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

# **Code Explanation and Logic**

This program simulates a **cinema ticket booking system** for a **5x5 seating arrangement**. It allows users to choose available seats, mark them as booked, and ensures that no seat is double-booked. It also validates user inputs for seat selection and ensures the system is responsive.

**Key Concepts and Technical Terms Used**

1. **2D Array (seats[5][5])**
   * The seating arrangement is stored in a **2D array** called seats, where each element is a character:
     + 'O' represents an **available seat**.
     + 'X' represents a **booked seat**.
   * The array is initialized with all seats as available ('O').
2. **User Input Handling**
   * The program repeatedly prompts the user to select a seat by entering a row and column number. The row and column are expected to be in the range of 1 to 5.
3. **Displaying the Seating Arrangement**
   * The program prints the current seating arrangement on the screen with rows and columns. Each seat is represented by either 'O' or 'X'.
4. **Booking a Seat**
   * After displaying the seating arrangement, the program asks for the user's desired row and column for booking.
   * The row and column are **converted to zero-based indexing** (subtracting 1 from the user input) for accessing the array elements.
5. **Input Validation**
   * **Invalid Row/Column Check**: If the user inputs a row or column outside the range 1 to 5, an error message is displayed.
   * **Already Booked Seat Check**: If the user selects a seat that is already booked ('X'), an error message is displayed.
6. **Updating the Seating Arrangement**
   * If the selected seat is available, it is marked as 'X', indicating it is booked.
7. **Loop and Exit Condition**
   * After each booking, the program asks if the user wants to book more seats. If the user inputs 'n' or 'N', the program exits.
8. **User Interaction**
   * The program continues to display the seating arrangement, ask for row and column input, and update the seating until the user decides to stop booking.

# Output

A computer screen shot of a black screen

AI-generated content may be incorrect.

# **Conclusion**

In this coursework, I worked on fundamental concepts of C++, focusing on data types, control structures, and arrays. I successfully implemented programs that required understanding and applying these concepts. Here's a brief summary of the tasks covered:

1. **Temperature Conversion Program**: I created a program that converts temperatures between Celsius and Fahrenheit using conditional statements. This task helped solidify my understanding of if-else conditions and the use of basic operators.
2. **Number Guessing Game**: I developed a number guessing game with difficulty levels using the switch-case structure and the rand() function. This allowed me to practice using loops, random number generation, and control structures effectively.
3. **Bouncy Number Checker**: I implemented a program to determine whether a number is "bouncy" based on its digit sequence. The task required using loops, conditional checks, and understanding how to manipulate integers and arrays.
4. **Cinema Ticket Booking System**: I built a cinema ticket booking system with a 5x5 seating arrangement, where users can book available seats and see real-time updates. This task helped me practice arrays, loops, user input validation, and conditional checks to manage the seat booking process.

**Skills Gained:**

* **Control Structures**: Mastering if-else, switch-case, and loops allowed me to handle various conditions and user inputs.
* **Data Types and Arrays**: I worked extensively with arrays and data types, especially when managing seat bookings and number operations.
* **Problem-Solving**: These tasks helped me enhance my logical thinking and problem-solving skills by implementing real-world applications like the cinema booking system and number analysis.

Overall, this coursework provided a solid foundation in C++ programming by combining theoretical concepts with practical applications. I feel more confident in applying control structures, arrays, and basic programming principles to solve different types of problems. The hands-on approach has been instrumental in reinforcing my understanding of these core topics.