Weekly Homework 1

March 19, 2025

1 Introduction

1.1 Basic Syntax & Data Types

1. Input/Output (`cin` , `cout`)
2. Variables and constants
3. Data types (`int` , `char` , `float` , `double` , `bool` ...)
4. Operators (`+` , `-` , `*` , `/` , `%` , `++` , `--` ...)

1.2 Control Flow (Loops & Conditional Statements)

1. if, if-else, switch-case
2. Loops: for, while, do-while
3. break and continue

Example: Using `if-else`

```cpp
#include <iostream>
using namespace std;

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    if (num % 2 == 0) {
        cout << num << " is even." << endl;
```

```
    } else {
        cout << num << " is odd." << endl;
    }
    return 0;
}
```

Example: Using `switch` Statement

```
#include <iostream>
using namespace std;

int main() {
    int day;
    cout << "Enter day number (1-3): ";
    cin >> day;

    switch (day) {
        case 1: cout << "Monday" << endl; break;
        case 2: cout << "Tuesday" << endl; break;
        case 3: cout << "Wednesday" << endl; break;
        default: cout << "Invalid day" << endl;
    }
    return 0;
}
```

Example: Using `while` Loop

```
#include <iostream>
using namespace std;

int main() {
    int num = 1;
    while (num <= 5) {
        cout << num << " ";
        num++;
    }
    return 0;
}
```

Example: Using `do-while` Loop

```cpp
#include <iostream>
using namespace std;

int main() {
    int num = 1;
    do {
        cout << num << " ";
        num++;
    } while (num <= 5);
    return 0;
}
```

1.3 Function

1. Function declaration and definition
2. Function parameters and return types

```cpp
#include <iostream>
using namespace std;

// Function to add two numbers
int add(int a, int b) {
    return a + b;
}
```

1.4 Arrays & Strings

Example: Array Traversal

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int n = sizeof(arr) / sizeof(arr[0]); // Get array size

    cout << "Array elements: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";  // Output: 10 20 30 40 50
    }
    return 0;
}
```

Example: String Manipulation

```cpp
#include <iostream>
#include <string> // Include string library
using namespace std;

int main() {
    string name = "Alice";
    cout << "Original: " << name << endl;

    name.append(" Wonderland"); // Append text
    cout << "Updated: " << name << endl; // Output: Alice Wonderland

    cout << "Length: " << name.length() << endl; // Get string length
    return 0;
}
```

1.5 Pointers & References

1. Basics of pointers (`*` , `&` , `-¿`)
2. Pointer arithmetic (ptr++, ptr–)
3. new and delete (Dynamic Memory Allocation)
4. Pointers to arrays, functions, and objects

Example: Pointers in C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 10;
    int *ptr = &x; // Pointer storing the address of x

    cout << "Value of x: " << x << endl;      // Output: 10
    cout << "Address of x: " << &x << endl;  // Memory address
    cout << "Pointer value: " << ptr << endl; // Address stored in ptr
    cout << "Value using pointer: " << *ptr << endl; // Dereferencing: 10
    return 0;
}
```

2 Git & GitHub

2.1 Install Git

Download Git from git-scm.com and install it.

2.2 Set Up Git

Before using Git, configure your name and email:

```
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"
```

2.3 Create a GitHub Account

Go to GitHub and sign up for a free account.

2.4 Create a New Repository on GitHub

1. Click on New Repository
2. Give it a name (e.g., "MyFirstRepo")
3. Select Public or Private
4. Click Create Repository

2.5 Initialize Git Locally

Open your terminal and run:

```
mkdir MyFirstRepo  # Create a new folder
cd MyFirstRepo     # Move into the folder
git init           # Initialize Git
```

2.6 Create and Add a File

```
echo "Hello, GitHub!" > README.md  # Create a file
git add README.md  # Add the file to the staging area
git commit -m "First commit"  # Save the changes
```

2.7 Connect to GitHub and Push the File

Copy the repository URL from GitHub, then run:

```
git remote add origin https://github.com/your-username/MyFirstRepo.git
git branch -M main  # Rename the default branch to main
git push -u origin main  # Push to GitHub
```

You can learn more about Git and GitHub in this article: Introduction to Git and GitHub (FreeCodeCamp).

3 Recursion

3.1 What is Recursion?

3.2 Why Recursion?

3.3 Recursion and Memory

3.4 Recursion versus Iteration

3.5 Notes on Recursion

1. Recursive algorithms have two types of cases, recursive cases and base cases.
2. Every recursive function case must terminate at a base case.
3. Generally , iterative solutions are more efficient than recursive solutions [due to the overhead of function calls].
4. A recursive algorithm can be implemented without recursive function calls using a stack, but it's usually more trouble than its worth. That means any problem that can be solved recursively can also be solved iteratively .
5. For some problems, there are no obvious iterative algorithms.
6. Some problems are best suited for recursive solutions while others are not.

3.6 Problems & Solutions

3.6.1 Towers of Hanoi puzzle.

```
#include <iostream>
using namespace std;

// Recursive function to solve Tower of Hanoi
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod) {
    if (n == 1) {
```

```
        cout << "Move disk 1 from " << from_rod << " to " << to_rod << endl;
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from " << from_rod << " to " << to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

int main() {
    int n = 3; // Number of disks
    towerOfHanoi(n, 'A', 'C', 'B');
    return 0;
}
```

3.6.2 Given an array , check whether the array is in sorted order with recursion.

```
#include <iostream>
using namespace std;

// Function to check if an array is sorted using recursion
bool isSorted(int arr[], int n) {
    // Base case: If there is only one or zero elements, it is sorted
    if (n == 1 || n == 0)
        return true;

    // If first element is greater than the second, it's not sorted
    if (arr[0] > arr[1])
        return false;

    // Recursive call to check the rest of the array
    return isSorted(arr + 1, n - 1);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5}; // Example sorted array
    int n = sizeof(arr) / sizeof(arr[0]);

    if (isSorted(arr, n))
        cout << "The array is sorted.\n";
    else
        cout << "The array is NOT sorted.\n";

    return 0;
}
```

4 Backtracking

Backtracking is an algorithmic technique for solving problems by exploring possible solutions incrementally. If a choice leads to a dead end, the algorithm backtracks to a previous decision point and tries another path. It is commonly used for solving puzzles, searching paths, and constraint-based problems like Sudoku and the N-Queens problem.

```cpp
#include <iostream>

using namespace std;

const int N = 8;       // N-Queens size
int board[N][N] = {0}; // 8x8 Chessboard
int solutions = 0;     // Count valid solutions

// Function to check if a queen can be placed at board[row][col]
bool isSafe(int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        if (board[i][col] == 1)
            return false; // Check column

        if (col - (row - i) >= 0 && board[i][col - (row - i)] == 1)
            return false; // Check left diagonal

        if (col + (row - i) < N && board[i][col + (row - i)] == 1)
            return false; // Check right diagonal
    }
    return true;
}

// Backtracking function to place queens
void solveNQueens(int row)
{
    if (row == N)
    { // All queens placed successfully
        solutions++;
        return;
    }
    for (int col = 0; col < N; col++)
    {
        if (isSafe(row, col))
        {
            board[row][col] = 1;   // Place queen
            solveNQueens(row + 1); // Recur for next row
            board[row][col] = 0;   // Backtrack
```

```
        }
    }
}

int main()
{
    solveNQueens(0);
    cout << solutions << endl;
    return 0;
}
```

5 Exercises

Exercise 1: Fibonacci Series

Problem Statement

Write a recursive function to compute the **nth Fibonacci number**. The Fibonacci series is defined as:

$$F(n) = F(n-1) + F(n-2)$$

where:

$$F(0) = 0, \quad F(1) = 1$$

C++ Function Signature

```
int fibonacci(int n);
```

Example Input & Output

```
Input: 5
Output: 0 1 1 2 3
```

Exercise 2: Factorial of a Number

Problem Statement

Write a recursive function to compute the **factorial** of a given number n. Factorial is defined as:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

where:

$$0! = 1$$

C++ Function Signature

```cpp
int factorial(int n);
```

Example Input & Output

```
Input: 5
Output: 120
```

Exercise 3: Generate All Binary Strings

Problem Statement

Write a recursive function to generate **all binary strings** of length n. A binary string consists only of '0's and '1's.

C++ Function Signature

```cpp
void generateBinaryStrings(int n, string str);
```

Example Input & Output

```
Input: 3
Output:
000
001
010
011
100
101
110
111
```

Exercise 4: Towers of Hanoi puzzle

Exercise 5: Given an array , check whether the array is in sorted order with recursion.

Exercise 6: N-Queens problem

Submission Rules

Students must adhere to the following submission guidelines:

1. Each solution must be submitted in a separate file:
   - `ex1.cpp` for the Fibonacci Exercise 1.
   - `ex2.cpp` for the Factorial Exercise 2.
2. The program should read input from **standard input** (`cin`) and output results to **standard output** (`cout`).
3. Code should be well-structured with proper indentation and comments explaining the logic.
4. Use **only recursion** to solve the problems. Iterative solutions will receive zero points.
5. The submission must be in a **compressed zip file** named **MSSV.zip**, containing:
   - The required C++ files. (ex1.cpp, ex2.cpp, ex3.cpp, ex4.cpp, ex5.cpp, etc.).
   - A `report.pdf` file describing the approach used in each solution.
6. Example Input/Output Format:
   - **Input:**

     ```
     5
     ```

   - **Output for Fibonacci:**

     ```
     0 1 1 2 3
     ```