C++ Assignments | Recursion - 4 | Week 11

1.Given an integer array containing unique numbers, return power set, containing all the subsets of
the set. [Leetcode 78]
2.Given an integer array which may contain duplicate numbers, return power set, containing all the
subsets of the set. [Leetcode 90]
3.Given a string, find the length of the longest common substring from two given strings.
4.Program to find the factorial of a given number.
5.Program to convert a decimal number to binary.

1.

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

// Function to generate power set of unique numbers
vector<vector<int>> subsets(vector<int>& nums) {
    vector<vector<int>> result;
    vector<int> subset;

    // Helper function to generate subsets recursively
    function<void(int)> generate_subsets = [&](int index) {
        if (index == nums.size()) {
            result.push_back(subset);
            return;
        }
        // Exclude current element
        generate_subsets(index + 1);
        // Include current element
        subset.push_back(nums[index]);
        generate_subsets(index + 1);
        subset.pop_back();
    };

    generate_subsets(0);
    return result;
}

int main() {
```

```
    vector<int> nums = {1, 2, 3};
    vector<vector<int>> result = subsets(nums);

    // Print all subsets
    cout << "All subsets:" << endl;
    for (auto subset : result) {
        cout << "[";
        for (int num : subset) {
            cout << num << " ";
        }
        cout << "]" << endl;
    }

    return 0;
}
```

2.

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

// Function to generate power set of numbers with duplicates
vector<vector<int>> subsetsWithDup(vector<int>& nums) {
    vector<vector<int>> result;
    vector<int> subset;

    sort(nums.begin(), nums.end()); // Sort to handle
duplicates

    // Helper function to generate subsets recursively
    function<void(int)> generate_subsets = [&](int index) {
        result.push_back(subset);
        for (int i = index; i < nums.size(); ++i) {
            if (i == index || nums[i] != nums[i - 1]) { //
Skip duplicates
                subset.push_back(nums[i]);
                generate_subsets(i + 1);
                subset.pop_back();
            }
        }
    };

    generate_subsets(0);
```

```
    return result;
}

int main() {
    vector<int> nums = {1, 2, 2};
    vector<vector<int>> result = subsetsWithDup(nums);

    // Print all subsets
    cout << "All subsets with duplicates:" << endl;
    for (auto subset : result) {
        cout << "[";
        for (int num : subset) {
            cout << num << " ";
        }
        cout << "]" << endl;
    }

    return 0;
}
```

3.

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

// Function to find length of longest common substring
int longestCommonSubstring(string s1, string s2) {
    int m = s1.size();
    int n = s2.size();
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));
    int max_len = 0;

    for (int i = 1; i <= m; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                max_len = max(max_len, dp[i][j]);
            } else {
                dp[i][j] = 0; // Reset for non-matching
characters
            }
        }
```

```
    }

    return max_len;
}

int main() {
    string s1 = "abcdef";
    string s2 = "xbcde";

    int length = longestCommonSubstring(s1, s2);
    cout << "Length of longest common substring: " << length
<< endl;

    return 0;
}
```

4.

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

// Function to calculate factorial of a number
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * factorial(n - 1);
}

int main() {
    int n = 5;
    cout << "Factorial of " << n << " is: " << factorial(n)
<< endl;
    return 0;
}
```

5.

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

// Function to convert decimal to binary
long long decimalToBinary(int n) {
```

```
    if (n == 0) {
        return 0;
    }
    return n % 2 + 10 * decimalToBinary(n / 2);
}

int main() {
    int n = 23;
    cout << "Binary representation of " << n << " is: " <<
decimalToBinary(n) << endl;
    return 0;
}
```