DATA STRUCTURES LABORATORY

Assignment 5

Name: **Katkar Prathamesh Shivaji**

Enrollment No.: 18114038

Batch : O2

Submission Date: 02 October 2019

Branch: CSE

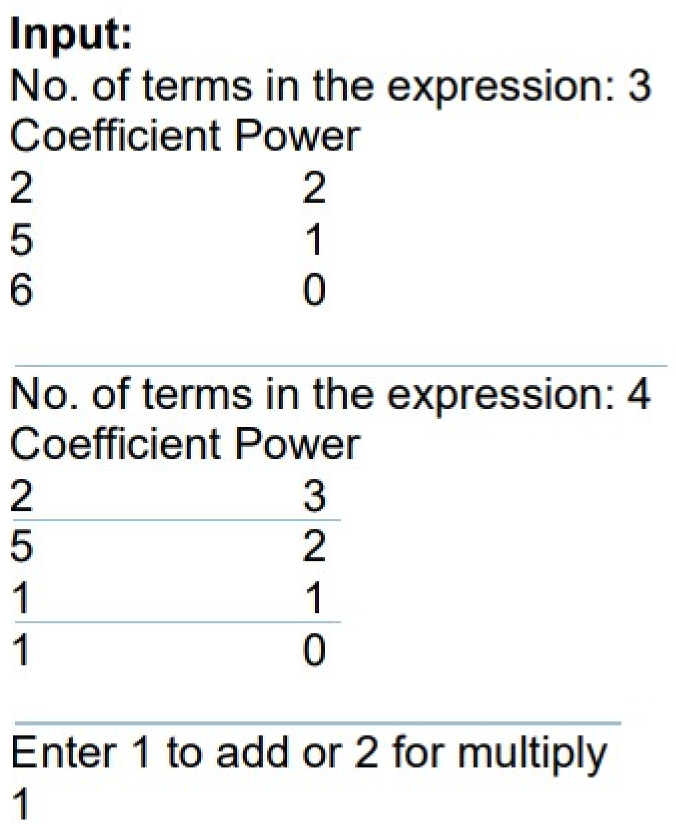
Email ID:

1. kshivaji@cs.iitr.ac.in (Piazza)

2. prathameshkatkar11@gmail.com (Moodle)

Problem Statement 1

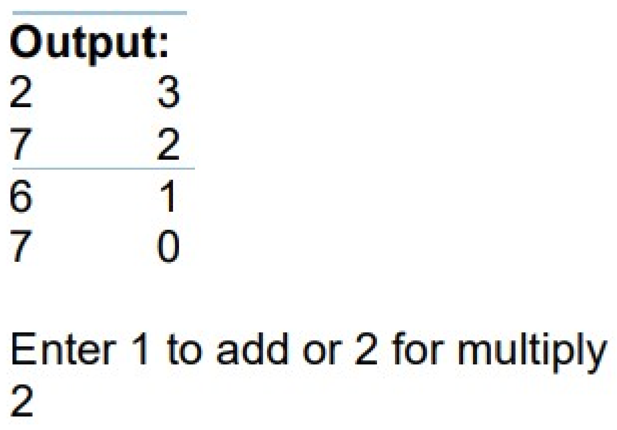
Write a C++ program to perform addition and multiplication of two polynomial expressions using any data structure chosen from STL. The polynomial expressions are of the form ax2 + bx + c, where a, b and c are real constants. The inputs for 2x2 + 5x + 6 and 2x3 + 5x2 + 1x + 1 are shown below (real constants followed by their power of x).



```
Input:
No. of terms in the expression: 3
Coefficient Power
2               2
5               1
6               0

No. of terms in the expression: 4
Coefficient Power
2               3
5               2
1               1
1               0

Enter 1 to add or 2 for multiply
1
```



```
Output:
2       3
7       2
6       1
7       0

Enter 1 to add or 2 for multiply
2
```

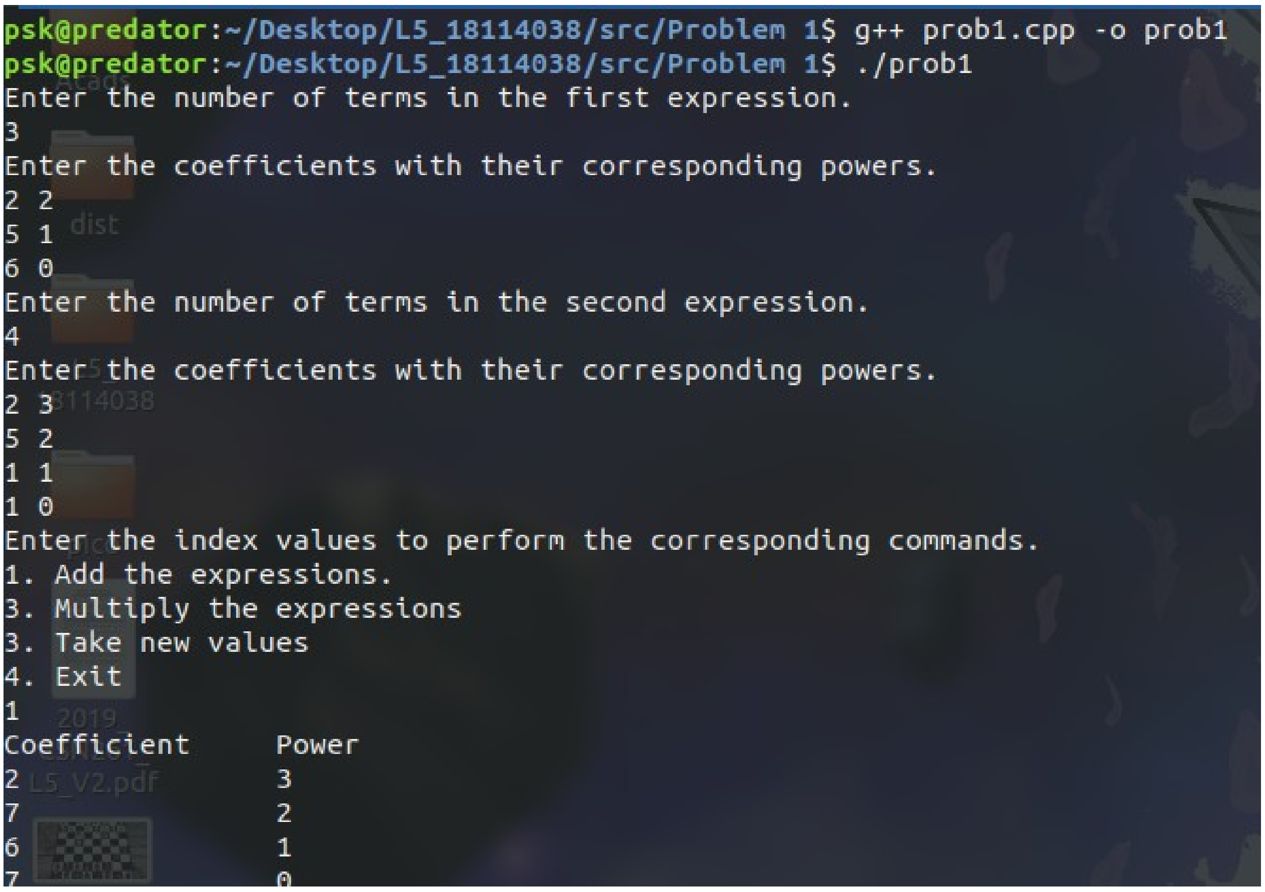
Output:

```
4    5
20   4
39   3
37   2
11   1
6    0
```

Data Structures used:

1. Ordered map (from STL)

Snapshots of running Code:



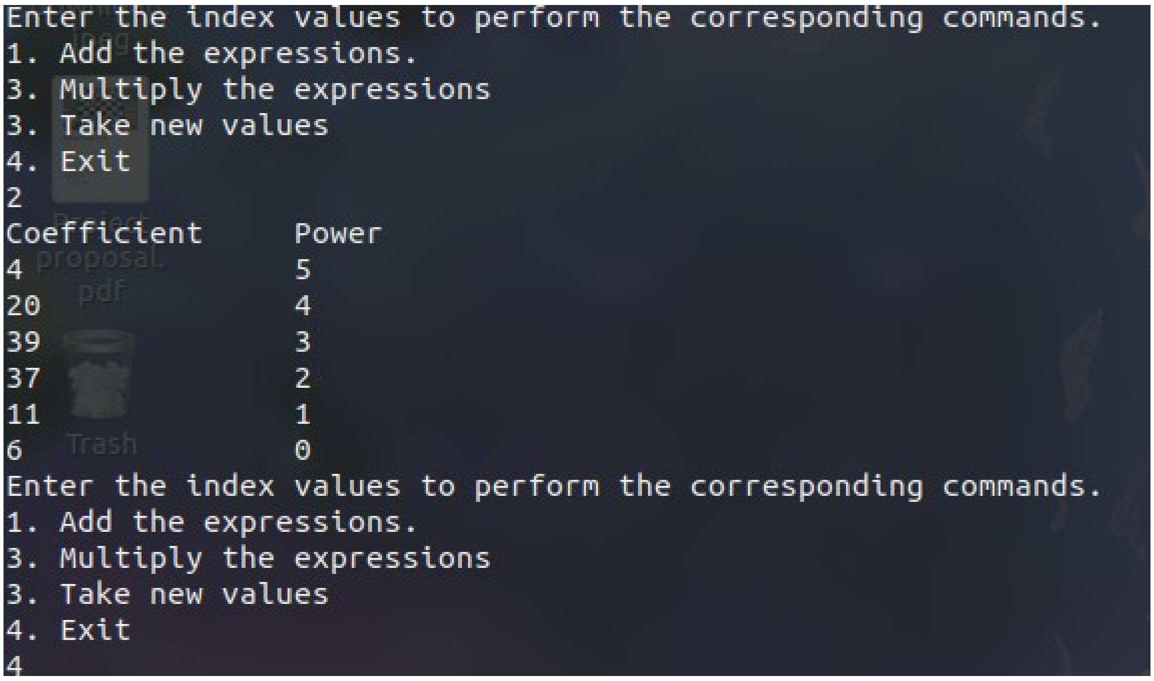
```
psk@predator:~/Desktop/L5_18114038/src/Problem 1$ g++ prob1.cpp -o prob1
psk@predator:~/Desktop/L5_18114038/src/Problem 1$ ./prob1
Enter the number of terms in the first expression.
3
Enter the coefficients with their corresponding powers.
2 2
5 1
6 0
Enter the number of terms in the second expression.
4
Enter the coefficients with their corresponding powers.
2 3
5 2
1 1
1 0
Enter the index values to perform the corresponding commands.
1. Add the expressions.
3. Multiply the expressions
3. Take new values
4. Exit
1
Coefficient     Power
2               3
7               2
6               1
7               0
```

Running time of code:

real 0m49.871s

user 0m0.005s

sys 0m0.000s



```
Enter the index values to perform the corresponding commands.
1. Add the expressions.
3. Multiply the expressions
3. Take new values
4. Exit
2
Coefficient     Power
4               5
20              4
39              3
37              2
11              1
6               0
Enter the index values to perform the corresponding commands.
1. Add the expressions.
3. Multiply the expressions
3. Take new values
4. Exit
4
```

<u>**Problem Statement 2**</u>

Given a set of nodes connected to each other in the form of a weighted undirected graph G, find the minimum spanning tree (MST). A spanning tree T of an undirected graph G is a subgraph that is a tree which includes all of the vertices of G, with minimum possible number of edges. G may have more than one spanning trees. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree. A minimum spanning tree (MST) is a spanning tree whose weight is less than or equal to that of every other spanning tree. For given input graph (given as a CSV file having the format as shown in the example below), implement Kruskal’s algorithm in C++ program using UNION FIND data structures (without using STL) and show all the edges of the MST as output in both the command line and in the “dot file”, where DOT is a graph description language. Also, print the total edge weight of the MST. For more details follow this link https://www.graphviz.org/doc/info/lang.html. Further use the “dot file” file to visualize the output graph in .pdf or .png file using Graphviz.

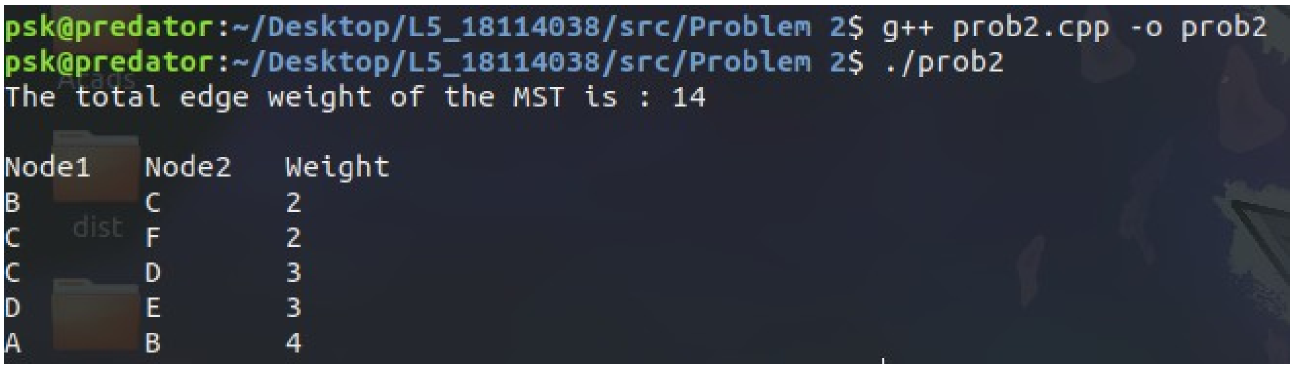
Alorithms Used:

1. Kruskal’s algorithm
2. Union by Rank and Path Compression
3. Inorder insertion in Linked List

Data Structures used:

1. Resizable Array
2. Linked List

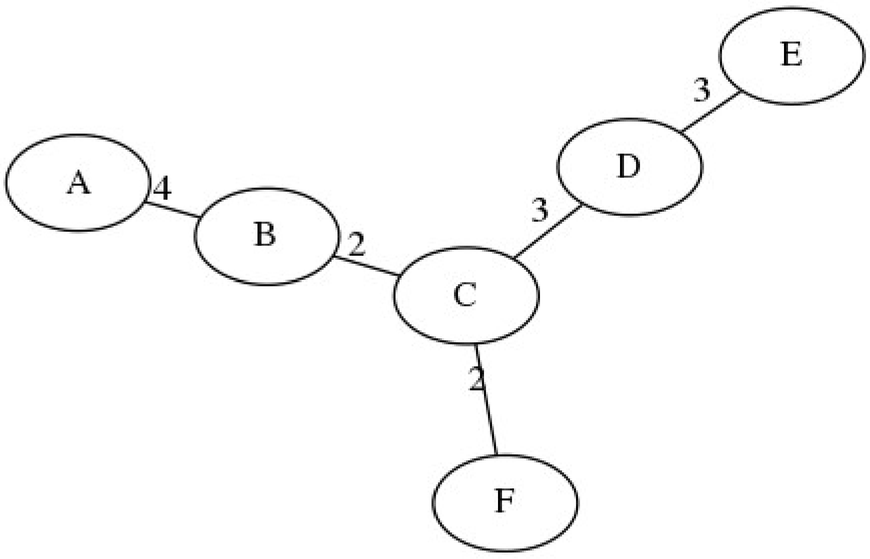
Snapshots of Code:



```
psk@predator:~/Desktop/L5_18114038/src/Problem 2$ g++ prob2.cpp -o prob2
psk@predator:~/Desktop/L5_18114038/src/Problem 2$ ./prob2
The total edge weight of the MST is : 14

Node1    Node2    Weight
B        C        2
C        F        2
C        D        3
D        E        3
A        B        4
```

Generated MST:



Running time of code:

```
real  0m0.067s
user  0m0.057s
sys   0m0.010s
```

Problem Statement 3

Write a C++ program to implement Prim's algorithm for a given input graph (given as a CSV file having the format as shown in the example below) using Fibonacci heap data structure to find the minimum spanning tree (MST). You can use STL for the data structure used in this C++ program. It is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm generates the MST by adding one vertex at a time, starting from an arbitrary vertex. At each step the cheapest possible edge weight is chosen from the already selected vertex. These algorithms find the minimum spanning forest in a possibly disconnected graph; in contrast, the most basic form of Prim's algorithm only finds minimum spanning tree in connected graphs. Show all the edges of the MST as the output in command line. Also, print the total edge weight of the MST. Use Newick file format (https://en.wikipedia.org/wiki/Newick_format) for visualization of the MST in ETE Toolkit (http://etetoolkit.org/)

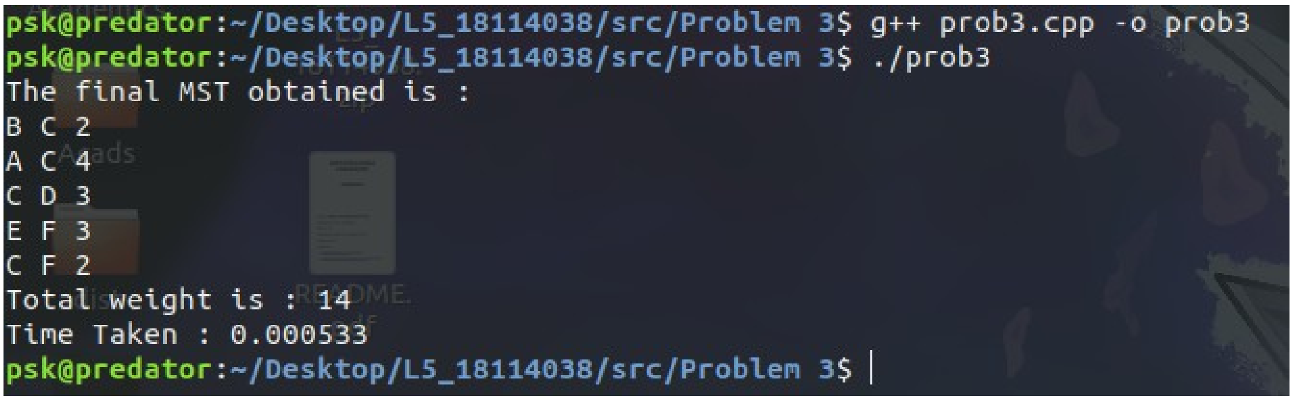
Data Structures used:

1. Fibonnaci Heap

Algorithms used:

1. Prim’s Algorithm

Snapshots of running code:



```
psk@predator:~/Desktop/L5_18114038/src/Problem 3$ g++ prob3.cpp -o prob3
psk@predator:~/Desktop/L5_18114038/src/Problem 3$ ./prob3
The final MST obtained is :
B C 2
A C 4
C D 3
E F 3
C F 2
Total weight is : 14
Time Taken : 0.000533
psk@predator:~/Desktop/L5_18114038/src/Problem 3$
```