**Worksheet-2.1**

**Student Name:-** Pushpraj Roy **UID:-** 20BCS9866

**Branch:-** BE- CSE **Section/Group:-** WM\_617 “A”

**Subjetct Code:-** 20CSP-312 **Semester:-** 5th

**Subject Name:-** DAA Lab

1. **Aim/Overview of the practical: -**

Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming

# Task to be done/ Which logistics used :-

Find an optimal solution to matrix chain multiplication using dynamic programming.

1. **Algorithm/Flowchart :-**
2. Start
3. Iterate from l = 2 to N-1 which denotes the length of the range:
4. Iterate from i = 0 to N-1:
5. Find the right end of the range (j) having l matrices.
6. Iterate from k = i+1 to j which denotes the point of partition.
7. Multiply the matrices in range (i, k) and (k, j).
8. This will create two matrices with dimensions arr[i-1]\*arr[k] and arr[k]\*arr[j].
9. The number of multiplications to be performed to multiply these two matrices (say X) are arr[i-1]\*arr[k]\*arr[j].
10. The total number of multiplications is dp[i][k]+ dp[k+1][j] + X.
11. The value stored at dp[1][N-1] is the required answer.
12. End
13. **Steps for experiment/practical/Code :-**

#include <iostream>

using namespace std;

bool isSubsetSum(int set[], int n, int sum)

{

if (sum == 0)

return true;

if (n == 0)

return false;

if (set[n - 1] > sum) return

isSubsetSum(set, n - 1, sum); return

isSubsetSum(set, n - 1, sum)

|| isSubsetSum(set, n - 1, sum - set[n - 1]);

}

int main()

{ int set[] = { 5, 10, 3, 9, 21, 55 }; int sum = 11;

int n = sizeof(set) / sizeof(set[0]);

if (isSubsetSum(set, n, sum) == true)

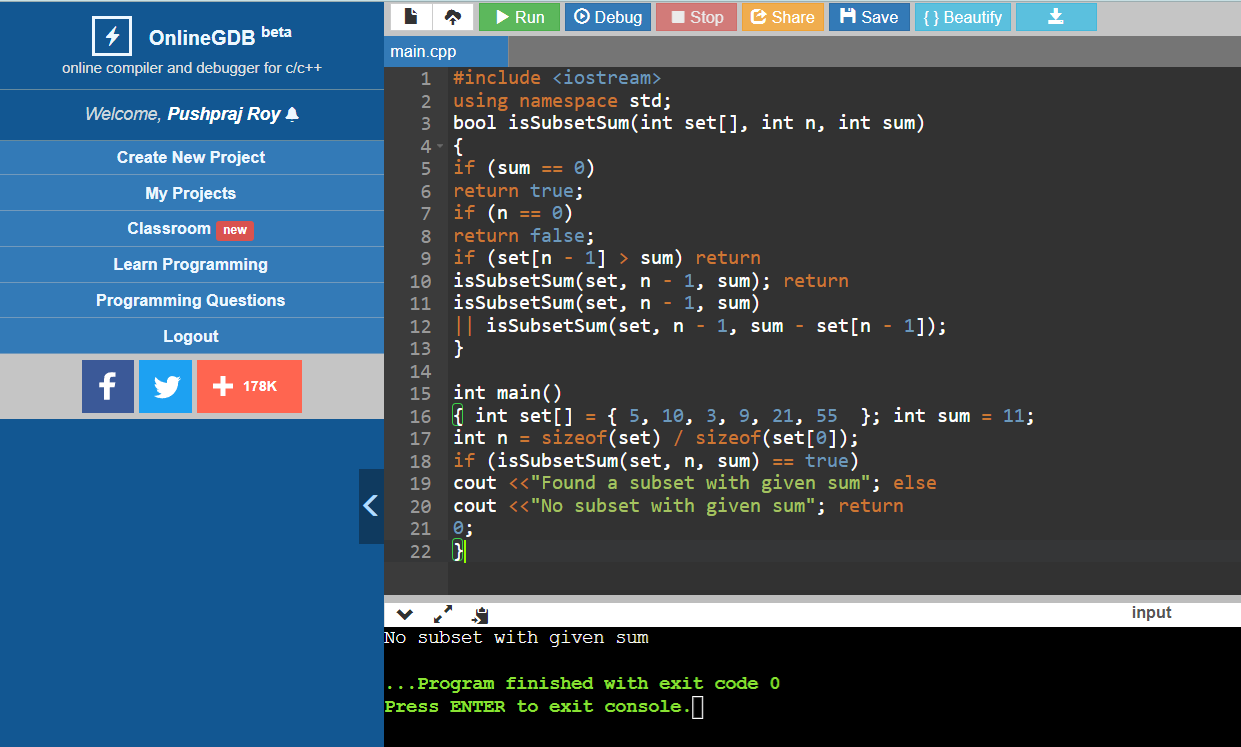
cout <<"Found a subset with given sum"; else

cout <<"No subset with given sum"; return

0;

}

1. **Result/Output :-**

****