**Worksheet-2.2**

**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: -**

To implement subset-sum problem using Dynamic Programming.

# Task to be done/ Which logistics used :-

find whether or not there exists any subset of the given set .

1. **Algorithm/Flowchart :-**
2. We create a boolean subset[][] and fill it in bottom up manner.
3. The value of subset[i][j] will be true if there is a subset of set[0..j-1] with sum equal to i., otherwise false.
4. Subset[i][j] = true if there is a subset with
5. The i-th element as the last element \* sum equal to j
6. Subset[i][0] = true as sum of {} = 0
7. Subset[0][j] = false as with no elements we can get no sum
8. Subset[i][j] = subset[i-1][j-E1]; where E1 = array[i-1]
9. Finally, we return subset[n][sum].
10. **Steps for experiment/practical/Code :-**

#include<iostream>

using namespace std;

bool subsetsum\_DP(int a[],int n, int sum)

{

bool dp[n+1][sum+1];

int i,j;

for(i=0;i<=n;i++)

dp[i][0]=true;

for(j=1;j<=sum;j++)

dp[0][j]=false;

for(i=1;i<=n;i++)

{

for(j=1;j<=sum;j++)

{

if(dp[i-1][j]==true)

dp[i][j]=true;

else

{

if(a[i-1]>j)

dp[i][j]=false;

else

dp[i][j]=dp[i-1][j-a[i-1]];

}

}

}

return dp[n][sum];

}

int main()

{

int set[] = { 3, 34, 4, 12, 5, 2 };

int sum = 9;

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

if (subsetsum\_DP(set, n, sum) == true)

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

else

cout <<"No subset with given sum";

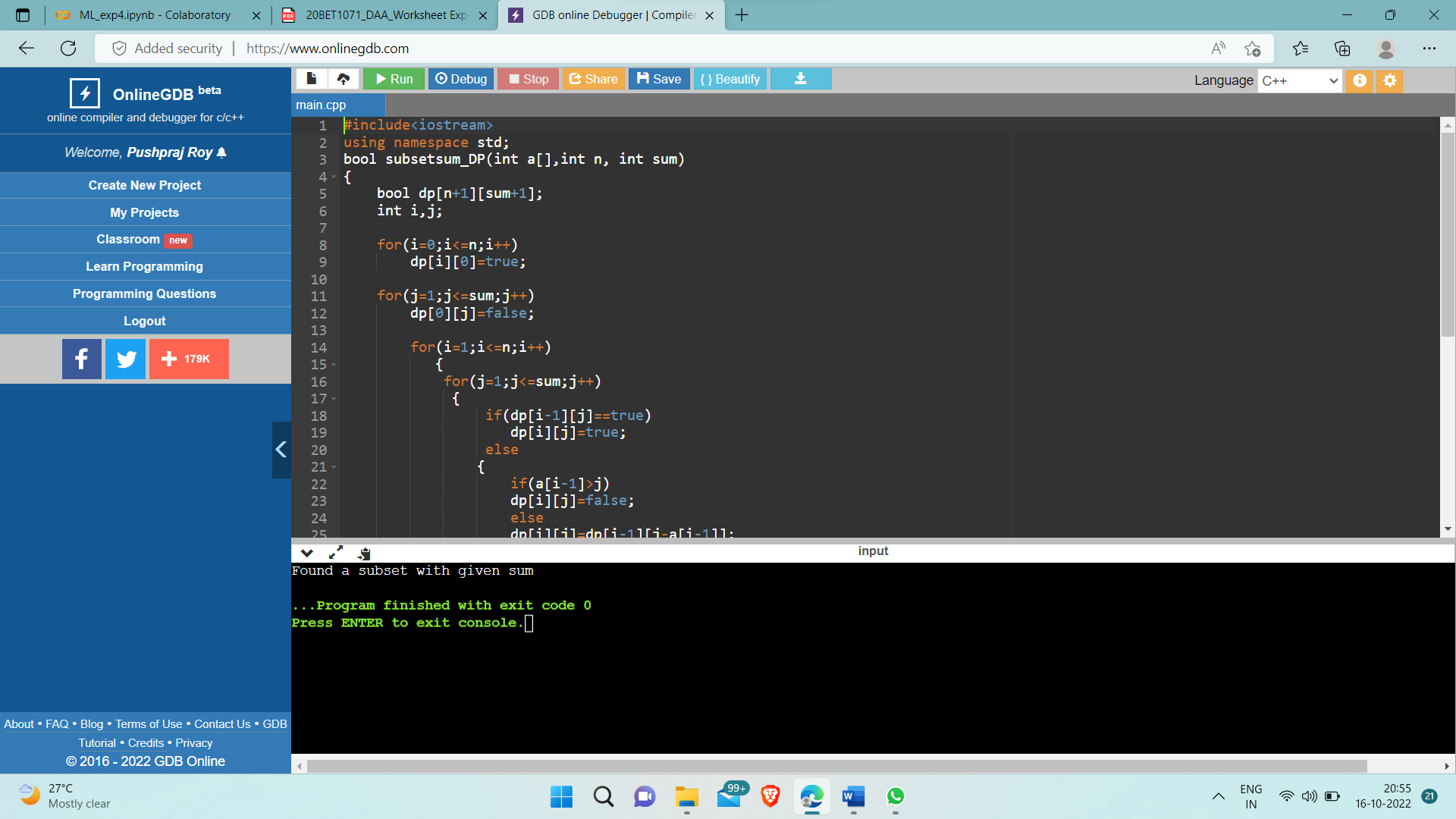
return 0;

}

1. **Observations/Discussions/ Complexity Analysis:**

* Worst case time complexity: Θ(n\*sum)
* Space complexity: Θ(sum)

1. **Result/Output :-**

****