Experiment 2.2

Student Name: Lipakshi **UID: 20BCS5082**

Branch: CSE **Section/Group: 20BCS-WM-607/B**

Semester: 5th **Date of Performance:16/09/2022**

Subject Name: DAA LAB **Subject Code: 21CSP-312**

1. Aim/Overview of the practical:

To implement a subset-sum problem using Dynamic Programming.

2. Task to be done/which logistics used:

To implement a subset sum problem using dynamic programming and problem-solving skills with the designing of the algorithm.

3. Algorithm/ Flowchart:

A naive solution would be to cycle through all subsets of n numbers and, for every one of them, check if the subset
sums to the right number. The running time is of order O(2n.n) since there are 2n subsets, and to check each subset, we need to sum at most n elements.
A better exponential-time algorithm uses recursion. Subset sum can also be thought of as a special case of the 0–1.

Knapsack problem. For each item, there are two possibilities:

1. Include the current item in the subset and recur for the remaining items with the remaining total.
2. Exclude the current item from the subset and recur for the remaining items.

Finally, return true if we get a subset by including or excluding the current item; otherwise, return false.
The recursion's base case would be when no items are left, or the sum becomes negative. Return true when the sum becomes 0, i.e., the subset is found.

4. Steps for experiment/practical/Code:

```
//SUBSET SUM
#include<bits/stdc++.h
> using namespace std;
bool sumSet(int* arr,int sum,int n)
{
	if (sum==0)
	{
		return true;
	}
	if (n==0 && sum!=0)
	{
		return false;
	}

	//if arr[i]>j//EXACT NEECHE COPY
	HOGA if (arr[n-1]>sum)
	{
		return sumSet(arr,sum,n-1);
	}
	//Reverse Checking
	//ACCORDING TO THE TOP ELEMENT IF TRUE THEN AS IT IS, IF FALSE THEN REVESAL TILL THE Nth ELEMENT.
	return sumSet(arr,sum,n-1)||sumSet(arr,sum-arr[n-1],n-1);
}
int main ()
{
	int size;
```

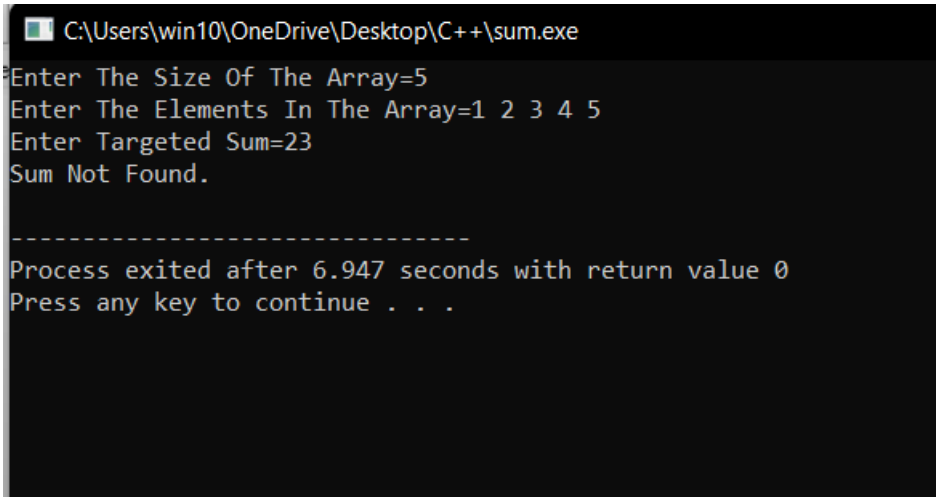
```
        cout<<"Enter The Size Of The
        Array="; cin>>size;
        int arr[size];
        cout<<"Enter The Elements In The
        Array="; for (int i=0;i<size;i++)
        {
                cin>>arr[i];
        }
        int sum;
        cout<<"Enter Targeted Sum=";
        cin>>sum;
        if (sumSet(arr,sum,size)==true)
        {
                                                cout<<"Sum Found."<<endl;

        }
        else
        {
                                                cout<<"Sum Not Found."<<endl;

        }

        return 0;
        getchar();
}
```

5. Output:



6. Observations/Discussions/ Complexity Analysis:

O(N*sum) where, N is the size of the array.
Space Complexity: O(N*sum).

7. Learning Outcomes:

a) Learned about dynamic programming.
b) Learned about design and algorithm analysis.
c) Learned about calculating time complexity of the subset sum problem using dynamic programming.
d) Learned about recursion.
e) Learned about initiating the base case of the condition.