Experiment – 6

Student Name: Ayush Ranjan
UID: 23BCS10187
Branch: BE-CSE
Section/Group: KRG-2-B
Semester: 5th
Date of Performance: 14/8/25
Subject Name: DAA
Subject Code: 23CSH-301

1. Aim:
Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.

2. Procedure:
- Start by defining the problem statement: determine if there exists a subset of a given set that adds up to a target sum.
- Take input for the number of elements, the set of integers, and the target sum.
- Initialize a 2D boolean DP table where each entry represents whether a subset with a given sum is possible.
- Fill the DP table iteratively using the relation that considers inclusion and exclusion of current elements.
- Check the final value in the DP table to determine if the subset with the given sum exists.
- Display the result and analyze the time and space complexity of the algorithm.

3. Code:

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

bool isSubsetSum(int set[], int n, int sum) {
    bool subset[n + 1][sum + 1];
    for (int i = 0; i <= n; i++)
        subset[i][0] = true;
    for (int i = 1; i <= sum; i++)
        subset[0][i] = false;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= sum; j++) {
            if (j < set[i - 1])
                subset[i][j] = subset[i - 1][j];
            else
                subset[i][j] = subset[i - 1][j] || subset[i - 1][j - set[i - 1]];
        }
    }

 return subset[n][sum];
```

```
}

int main() {
    int n, sum;
    cout << "Enter number of elements: ";
    cin >> n;
    int set[n];
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++) cin >> set[i];
    cout << "Enter target sum: ";
    cin >> sum;
    if (isSubsetSum(set, n, sum))
        cout << "Subset with given sum exists.";
    else
        cout << "No subset with given sum exists.";
    return 0;
}
```

4. Output:

```
Enter number of elements: 5
Enter elements: 3 34 4 12 5
Enter target sum: 9
Subset with given sum exists.
```

5. Learning Outcomes:

- Gained knowledge of solving problems using Dynamic Programming principles.
- Learned how to break a complex problem into overlapping subproblems.
- Developed understanding of memory optimization through tabulation.
- Learned to analyze time and space complexity of dynamic algorithms.
- Gained confidence in applying DP concepts to real-world computational problems.