EXPERIMENT NO.10

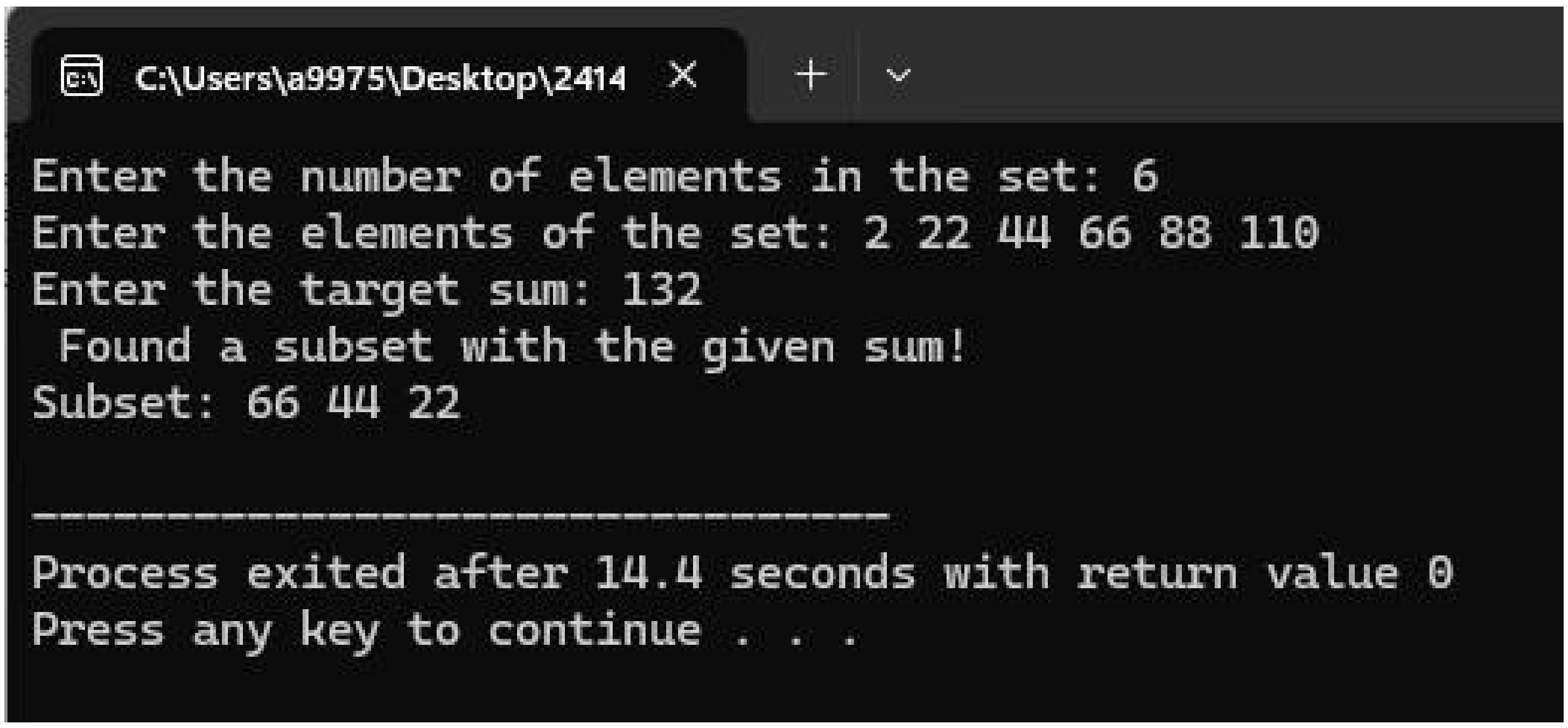
Sum Of Subset Problem

Reg. No.:-24141028

Program:-

```
#include<iostream>

#include<vector>

using namespace std;

bool isSubsetSumDP(vector<int>&set, int sum, vector<int>&subset) {

  int n = set.size();

  int i, j;

  vector<vector<bool>>dp(n + 1, vector<bool>(sum + 1, false));

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

    dp[i] [0] = true;

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

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

      if (set[i -  1] > j)

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

      else

        dp[i] [j] = dp[i -  1] [j] || dp[i -  1] [j -  set[i -  1]];

    }
    }
    if (!dp[n][sum])
        return false;
    for (i = n, j = sum; i>0&&j>0; --i) {
        if (!dp[i– 1][j]) {
            subset.push_back(set[i– 1]);
            j-= set[i– 1];
        }
    }
    return true;
}
int main() {
    int n,i, sum;
    cout<< “Enter the number of elements in the set: “;
    cin>>n;
    vector<int>set(n);
    cout<< “Enter the elements of the set: “;
    for (i = 0; i<n; ++i)
        cin>>set[i];
    cout<< “Enter the target sum: “;
    cin>>sum;
    vector<int>subset;

    if (isSubsetSumDP(set, sum, subset)) {

       cout<< "Found a subset with the given sum!\nSubset: ";

       for (i = 0; i<subset.size(); ++i)

          cout<<subset[i]<< " ";

       cout<<endl;

    }else{

       cout<< "? No subset found with the given sum." <<endl;

    }

    return 0;

}
```

Output:-



```
Enter the number of elements in the set: 6
Enter the elements of the set: 2 22 44 66 88 110
Enter the target sum: 132
 Found a subset with the given sum!
Subset: 66 44 22

--------------------------------
Process exited after 14.4 seconds with return value 0
Press any key to continue . . .
```

Algorithm:-

Input: Set of positive integers $S = \{s_1, s_2, \ldots, s_n\}$

Target sum d.

Output:- All subsets of S whose elements sum to d Step 1: Sort the set S (optional for optimization).

Step 2: Initialize variables:

sum = 0, k = 0 (starting index).

Step 3: Call subset(k, sum) recursively.

Step 4:

```
Subset(k, sum):
    1. If sum == d:
         Print current subset; return.
    2. If sum>d or k>=n:
         Return (backtrack).
    3. Include S[k] in subset and call subset(k + 1, sum + S[k]).
    4. Exclude S[k] and call subset(k + 1, sum).
```

Time Complexity: $O(2^n)$

Space Complexity: $O(n)$

List of Applications:-

1. Knapsack problem
2. Resource allocation
3. Budget planning
4. Cryptography
5. Load balancing
6. Data partitioning

7. Combinatorial optimization
8. Decision support systems
9. Scheduling tasks
10. Power set generation