Lab 6: Implementation of sum of subset problem using backtracking

Objective:

To implement the sum of subset problem using backtracking method.

Theory:

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece ate a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time. It is a form of recursive depth first search.

Algorithm:

I. Create a recursive function that takes the following parameters:
   A. The current subset
   B. The current sum
   C. The target sum
   D. The index of the current element being considered
II. If the current sum equals the target sum, we've found a valid subset.
III. If the current sum exceeds the target sum or we've considered all elements, backtrack
IV. For the current element, we have two choices:
   A. Include it in the subset
   B. Exclude it from the subset
V. Recursively try both choices

Observation:

```
#include <bits/stdc++.h>
using namespace std;

class subsetSum {
   public:
   bool flag = 0;
   void PrintSubsetSum(int i, int n, int set[], int targetSum,
                       vector<int>& subset)
   {
      if (targetSum == 0) {

         flag = 1;
         cout << "[ ";
         for (int i = 0; i < subset.size(); i++) {
            cout << subset[i] << " ";
         }
         cout << "]";
         return;
      }

      if (i == n) {
         return;
      }
      PrintSubsetSum(i + 1, n, set, targetSum, subset);

      if (set[i] <= targetSum) {

         subset.push_back(set[i]);

         PrintSubsetSum(i + 1, n, set, targetSum - set[i],
                  subset);

         subset.pop_back();
      }
   }
};

long long getTime(std::function<void()> f){
   auto start = clock();
   f();
   auto end = clock();
   long double duration = end - start;
   return (duration/CLOCKS_PER_SEC) * 1000000000;
}
```

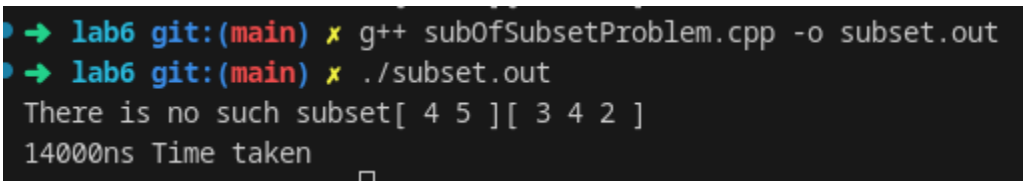
```
int main()
{
   subsetSum s;
   int set[] = { 3, 34, 4, 12, 5, 2 };
   int n = sizeof(set) / sizeof(set[0]);
   int targetSum = 9;
   vector<int> subset;
   auto subsetProblem = [&](){
      s.PrintSubsetSum(0, n, set, targetSum, subset);
      cout << endl;
   };

   if (!s.flag) {
      cout << "There is no such subset";
   }

   cout << getTime(subsetProblem) << "ns Time taken" << endl;

   return 0;
}
```

Output:



Conclusion:

We implemented backtracking algorithm to solve subset sum problem in C++.