RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 08
Course No.: CSE 2202

Date of Experiment: 21.01.2019
Date of Submission: 28.01.2019

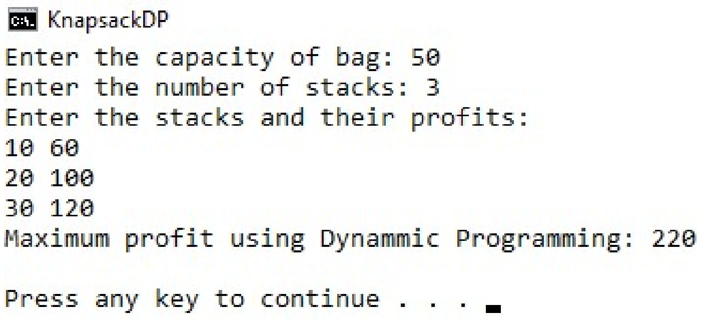
Submitted to:	**Submitted by:**
Biprodip Pal	Riyad Morshed Shoeb
Assistant Professor,	Roll No: 1603013
Department of Computer Science and Engineering	Section: A
Rajshahi University of Engineering and Technology	Department of Computer Science and Engineering
	Rajshahi University of Engineering and Technology

Problem-1: Given W_i and P_i for N objects, find the maximum profit using 0/1 knapsack using dynamic programming approach.

Code:

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


//class definition
class MoneyStack
{
public:
    int StackWeight;
    int StackProfit;
};


//function definition
int MAXIMUM(int a, int b)
{
    return (a>b)? a:b;
}

int   KnapsackDP(MoneyStack   Stacks[],   int   NumberOfStacks,   int 
CapacityOfBag)
{
    int iter;
    int weight;
    int TABLE[NumberOfStacks+1][CapacityOfBag+1];

    for(iter=0; iter<=NumberOfStacks; iter++)
    {
        for(weight=0; weight<=CapacityOfBag; weight++)
        {
            if(iter==0 || weight==0)
                TABLE[iter][weight] = 0;
            else if(Stacks[iter-1].StackWeight <= weight)
                TABLE[iter][weight]      =      MAXIMUM(Stacks[iter-
1].StackProfit+TABLE[iter-1][weight-Stacks[iter-1].StackWeight], 
TABLE[iter-1][weight]);
            else
                TABLE[iter][weight] = TABLE[iter-1][weight];
        }
    }

    return TABLE[NumberOfStacks][CapacityOfBag];
}


//main function
int main(void)
{
    int CapacityOfBag;
    int NumberOfStacks;
    int iter;

    cout << "Enter the capacity of bag: ";
    cin >> CapacityOfBag;
    cout << "Enter the number of stacks: ";
    cin >> NumberOfStacks;

    MoneyStack Stacks[NumberOfStacks];

    cout << "Enter the stacks and their profits:" << endl;
    for(iter=0; iter<NumberOfStacks; iter++)
        cin >> Stacks[iter].StackWeight >> Stacks[iter].StackProfit;

    cout  <<  "Maximum  profit  using  Dynammic  Programming:  "  <<
KnapsackDP(Stacks, NumberOfStacks, CapacityOfBag) << endl;
}
```

Output:



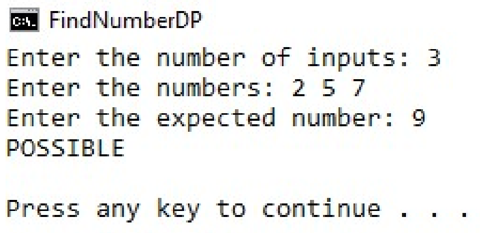
```
KnapsackDP
Enter the capacity of bag: 50
Enter the number of stacks: 3
Enter the stacks and their profits:
10 60
20 100
30 120
Maximum profit using Dynammic Programming: 220

Press any key to continue . . .
```

Problem-2: Given a set of numbers $i.e. \{2, 5, 7\}$ and an integer N $(i.e.\ 9)$, find if elements of the given set can make the total.

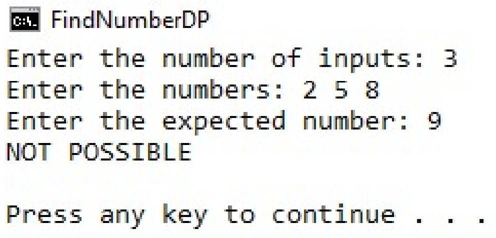
Code:

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <vector>
using namespace std;


//global variables
vector <int> GivenNumbers;


//function definition
bool is_possible(int RequiredNumber)
{
    int NumberOfInputs = GivenNumbers.size();
    int row_iter;
    int column_iter;
    bool POSSIBLE[NumberOfInputs+1][RequiredNumber+1];

    for(row_iter=0; row_iter<=NumberOfInputs; row_iter++)
    {
        for(column_iter=0;              column_iter<=RequiredNumber;
column_iter++)
        {
            if(column_iter == 0)
                POSSIBLE[row_iter][column_iter] = true;
            else if(row_iter == 0)
                POSSIBLE[row_iter][column_iter] = false;
            else if(column_iter <= GivenNumbers[row_iter-1])
                POSSIBLE[row_iter][column_iter]    =    ((true    &&
POSSIBLE[row_iter-1][column_iter  -  GivenNumbers[row_iter-1]])  ||
POSSIBLE[row_iter-1][column_iter]);
            else
                POSSIBLE[row_iter][column_iter] = POSSIBLE[row_iter-
1][column_iter];
        }
    }

    return POSSIBLE[NumberOfInputs][RequiredNumber];
}


//main function
int main(void)
{
    int NumberOfInputs;
    int RequiredNumber;
    int temp;
    int iter;
    cout << "Enter the number of inputs: ";
    cin >> NumberOfInputs;
    cout << "Enter the numbers: ";
    for(iter=0; iter<NumberOfInputs; iter++)
    {
        cin >> temp;
        GivenNumbers.push_back(temp);
    }
    cout << "Enter the expected number: ";
    cin >> RequiredNumber;

    if(is_possible(RequiredNumber))
        cout << "POSSIBLE" << endl;
    else
        cout << "NOT POSSIBLE" << endl;
}
```

Output:



```
Enter the number of inputs: 3
Enter the numbers: 2 5 7
Enter the expected number: 9
POSSIBLE

Press any key to continue . . .
```



```
Enter the number of inputs: 3
Enter the numbers: 2 5 8
Enter the expected number: 9
NOT POSSIBLE

Press any key to continue . . .
```