```
/*

DSC++ - ESE

Q1 - Coin Change Problem

Shashank Bagda - 92100133020

*/


#include <bits/stdc++.h>

using namespace std;


int count(int coins[], int n, int sum)

{


        // If sum is 0 then there is 1 solution

        // (do not include any coin)

        if (sum == 0)

                return 1;


        // If sum is less than 0 then no

        // solution exists

        if (sum < 0)

                return 0;


        // If there are no coins and sum

        // is greater than 0, then no

        // solution exist

        if (n <= 0)

                return 0;


        // count is sum of solutions (i)
        // including coins[n-1] (ii) excluding coins[n-1]
        return count(coins, n - 1, sum)
                + count(coins, n, sum - coins[n - 1]);
}

int main()
{
        int i, j, value, num, sum;

        printf("How many different coins you have? ");
        scanf("%d",&num);

        int coins[num] = {};    //Pre defined set of coins

        for(int lop=0; lop<num; lop++)
        {
                cout<<"Enter the "<< lop <<" value of the coin : ";
                cin>>coins[lop];
        }

        int n = sizeof(coins) / sizeof(coins[0]);

        cout<<"Enter the Target Value : ";
        cin>>sum;

        cout << "The total number of ways are : " << count(coins, n, sum);

        return 0;
}
```

```
/*

DSC++ - ESE

Q2 - Unsorted Array and Find Pair with given sum

Shashank Bagda - 92100133020

*/


#include <bits/stdc++.h>

using namespace std;


void findPair(int arr[], int n, int tar)

{

    int res_l, res_r;

            int l = 0, r = n-1, diff = INT_MAX;


    while (r > l)

    {

        if (abs(arr[l] + arr[r] - tar) < diff)

        {

            res_l = l;

            res_r = r;

            diff = abs(arr[l] + arr[r] - tar);

        }


        if (arr[l] + arr[r] > tar)

            r--;

        else

            l++;

    }
    cout <<"\nThe sum pair of the given target is " << arr[res_l] << " and " << arr[res_r];
}

int main()
{

            int range, tar, temp;

            cout<<"How many Integers are there in your Array : ";
            cin>>range;

            int arr[range] = {};

            for(int i=0; i<range; i++)          //Array for input is defined
            {
                    cout<<"Enter Integer of Index "<<i<<" : ";
                    cin>>arr[i];
            }

            cout<<"Enter the Target Value : ";                  //Target Integer
            cin>>tar;

            //Sorting the unsorted array
            for(int z=0; z<range; z++)
            {
                    for(int i=0; i<range; ++i)
                    {
                            for(int j=i+1; j<range; ++j)
                {
                        if(arr[i] > arr[i+1])
                        {
                            temp =  arr[i];
                    arr[i] = arr[i+1];
                    arr[i+1] = temp;
                        }
                            }
                    }
            }

            // Printing the Array
            printf("Sorted Array is : ");
            for(int i=0; i<range; i++)
            {
                    printf("%d ",arr[i]);
            }



            int n = sizeof(arr)/sizeof(arr[0]);
    findPair(arr, n, tar);

            return 0;
}
```

```
/*
DSC++ - ESE
Q3 - Given Linked List is in Palindrome
Shashank Bagda - 92100133020
*/

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



class Node
{
public:
	int data;
	Node(int d) { data = d; }
	Node* ptr;
};

// Function to check if the linked list is palindrome or not
bool Palindrome(Node* head)
{
	Node* slow = head;
	stack<int> s;	//Create Stack and push all elements

	while (slow != NULL)
	{
		s.push(slow->data);
		slow = slow->ptr;
	}
```

```
        while (head != NULL)
        {
                int i = s.top();
                s.pop();

                if (head->data != i)
                {
                        return false;
                }

                head = head->ptr;
        }

        return true;
}

int main()
{

        // Addition of linked list
        Node one    = Node(1);
        Node two    = Node(2);
        Node three  = Node(3);
        Node four   = Node(4);
        Node five   = Node(5);
        Node six    = Node(6);
        Node seven  = Node(6);
        Node eight  = Node(5);
        Node nine   = Node(4);
        Node ten    = Node(3);
        Node eleven = Node(2);
        Node twelve = Node(1);

        // Initialize the next pointer of every current pointer
        one.ptr = &two;
        two.ptr = &three;
        three.ptr = &four;
        four.ptr = &five;
        five.ptr = &six;
        six.ptr = &seven;
        seven.ptr = &eight;
        eight.ptr = &nine;
        nine.ptr = &ten;
        ten.ptr = &eleven;
        eleven.ptr = &twelve;
        twelve.ptr = NULL;

        Node* temp = &one;

        // Call function to check palindrome or not
        int result = Palindrome(&one);

        if (result == 1)
                cout << "Given Linked List is Palindrome";
        else
                cout << "Given Linked List is NOT Palindrome";
                return 0;
        }
```