Name:- Sushant Raj UID:- 22BCS10033 Section:- 22KPIT-901-A

DAY-3

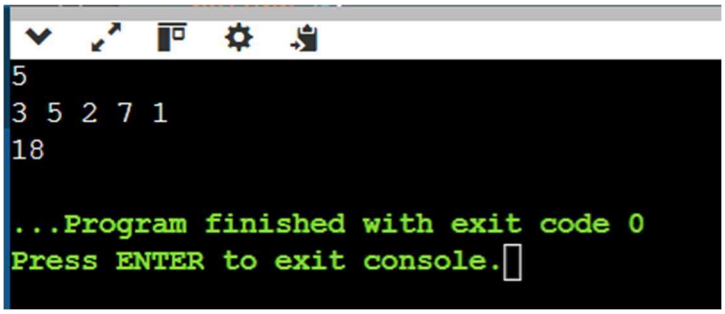
Q.1. Given an array of integers, find sum of array elements using recursion.

Program Code:-

```
#include <iostream> using

namespace std;


int sumArray(int arr[], int n) {     if (n ==

0) return 0;     return arr[n - 1] +

sumArray(arr, n - 1);

}


int main() {

   int n;

   cin >> n;     int arr[n];

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

{        cin >> arr[i];

   }
```

```
    cout << sumArray(arr, n);

return 0; }
```

Output:-



Q.2. You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

Program Code:-

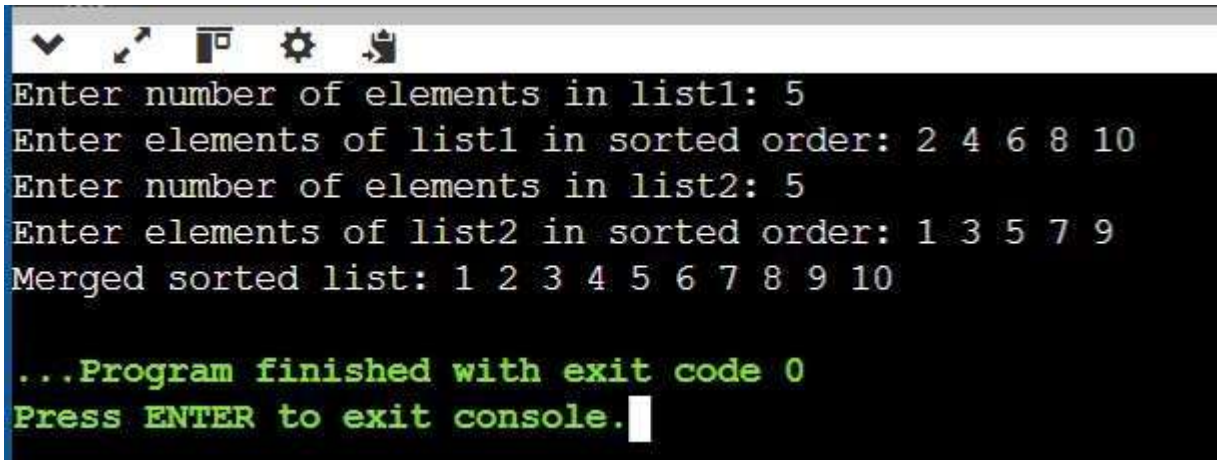
```
#include <iostream> using

namespace std; struct

ListNode {


    int val;

    ListNode* next;

    ListNode(int x) : val(x), next(nullptr) {}

};
```

```
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
   if (!list1) return list2;     if (!list2) return list1;     if
(list1->val <= list2->val) {         list1->next =
mergeTwoLists(list1->next, list2);         return
list1;     } else {         list2->next =
mergeTwoLists(list1, list2->next);         return list2;
   }
}

void printList(ListNode* head)
{     while (head != nullptr) {

       cout << head->val << " ";
head = head->next;
   }
}
ListNode* createList(int n) {
   ListNode* head = nullptr;
ListNode* tail = nullptr;
for (int i = 0; i < n; i++)
```

```
{        int val;        cin >>
val;
       ListNode* newNode = new ListNode(val);
       if (!head) {            head
= newNode;            tail =
newNode;        } else
{            tail->next =
newNode;            tail =
newNode;
       }
   }
   return head; } int main() {     int n1, n2;     cout
<< "Enter number of elements in list1: ";     cin >>
n1;     cout << "Enter elements of list1 in sorted
order: ";     ListNode* list1 = createList(n1);     cout
<< "Enter number of elements in list2: ";     cin >>
n2;     cout << "Enter elements of list2 in sorted
order: ";
   ListNode* list2 = createList(n2);
```

```
    ListNode* mergedList = mergeTwoLists(list1, list2);

cout << "Merged sorted list: ";

printList(mergedList);     return 0; }
```

Output:-



Q.3. You are given an integer array nums. Two players are playing a game with this array: player 1 and player 2.

Player 1 and player 2 take turns, with player 1 starting first. Both players start the game with a score of 0. At each turn, the player takes one of the numbers from either end of the array (i.e., nums[0] or nums[nums.length - 1]) which reduces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.

Return true if Player 1 can win the game. If the scores of both players are equal, then player 1 is still the winner, and you should also return true. You may assume that both players are playing optimally.

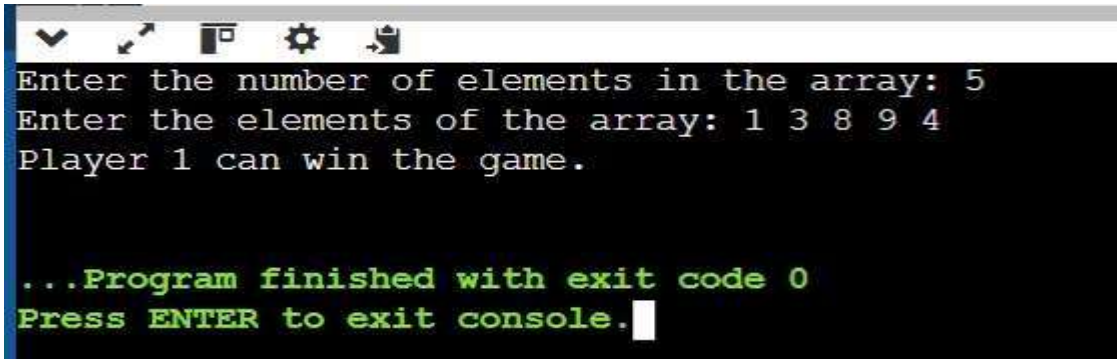
Program Code:-

```
#include <iostream> using namespace

std; bool canPlayer1Win(int nums[], int
```

```
n) {     int dp[n][n];     for (int i = 0; i <
n; i++) {         dp[i][i] = nums[i];

    }
    for (int length = 2; length <= n; length++) {         for (int i = 0; i
<= n - length; i++) {             int j = i + length - 1;             dp[i][j] =
max(nums[i] - dp[i + 1][j], nums[j] - dp[i][j - 1]);
        }
    }
    return dp[0][n - 1] >= 0;
} int main()
{
    int n;
    cout << "Enter the number of elements in the array: ";
cin >> n;     int nums[n];     cout << "Enter the elements
of the array: ";     for (int i = 0; i < n; i++) {         cin >>
nums[i];
    }
    if (canPlayer1Win(nums, n)) {
        cout << "Player 1 can win the game." << endl;
```

```
    } else {        cout << "Player 1 cannot win the

game." << endl;

    }

return 0; }
```

Output:-



Q.4. Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return the result of the evaluation.

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

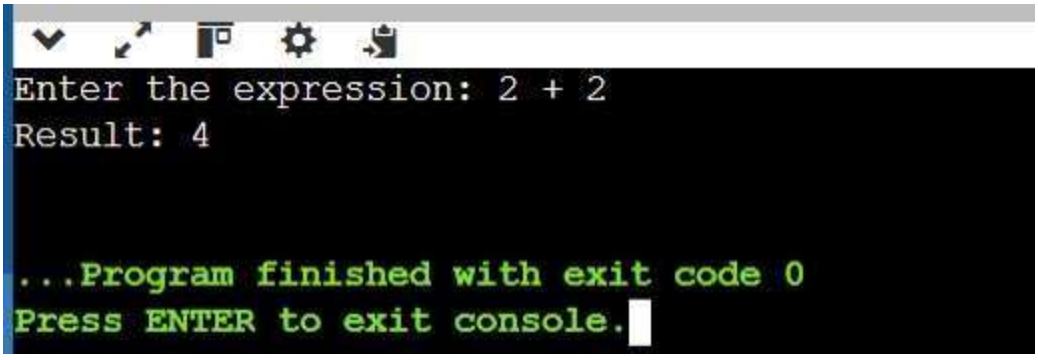
Program Code:-

```
#include <iostream>

#include <stack> using

namespace std; int

calculate(string s)

{     stack<int> nums,

ops;     int num = 0,
```

```
result = 0, sign = 1;
for (int i = 0; i <
s.size(); i++)
{       char c = s[i];
if (isdigit(c))
{           num = num *
10 + (c - '0');       }
else if (c == '+')
{           result += sign
* num;           num =
0;           sign =
1;       } else if (c ==
'-') {           result +=
sign * num;
num = 0;           sign
= -1;       } else if (c
== '('
{           nums.push(re
sult);
```

```
ops.push(sign);

result = 0;

            sign = 1;         } else if (c == ')')

{             result += sign * num;             num =

0;             result = nums.top() + ops.top() *

result;             nums.pop();             ops.pop();

        }

    }

    result += sign * num;     return result; } int

main() {     string s;     cout << "Enter the

expression: ";     getline(cin, s);     cout <<

"Result: " << calculate(s) << endl;     return

0;

}
```

Output:-



Q.5. You are given a positive integer primeFactors. You are asked to construct a positive integer n that satisfies the following conditions:

The number of prime factors of n (not necessarily distinct) is at most primeFactors.

The number of nice divisors of n is maximized. Note that a divisor of n is nice if it is divisible by every prime factor of n. For example, if n = 12, then its prime factors are [2,2,3], then 6 and 12 are nice divisors, while 3 and 4 are not.

Return the number of nice divisors of n. Since that number can be too large, return it modulo 109 + 7.

Note that a prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. The prime factors of a number n is a list of prime numbers such that their product equals n.

Program Code:-

```
#include <iostream>

#include <cmath> using

namespace std; const

int MOD = 1e9 + 7;

long long

powerMod(long long

base, long long exp,

long long mod)

{     long long result =

1;     while (exp > 0)
```

```
{        if (exp % 2 == 1)
{            result = (result
* base) % mod;
       }
       base = (base * base) % mod;
exp /= 2;
   }     return
result;
} int maxNiceDivisors(int primeFactors)
{     if (primeFactors == 1) return 1;
int q = primeFactors / 3;     int r =
primeFactors % 3;


   if  (r  ==  0)  {                  return
powerMod(3, q, MOD);      } else if
(r == 1) {        return (powerMod(3,
q - 1, MOD) * 4) % MOD;
   } else {
       return (powerMod(3, q, MOD) * 2) % MOD;
```

```
    }
} int main() {     int primeFactors;     cout <<
"Enter the number of prime factors: ";     cin >>
primeFactors;
    cout << "Number of nice divisors: " << maxNiceDivisors(primeFactors) <<
endl;     return 0; }
```

Output:-