Assignment 3

Name – Ishita   Faculty Name- Er. Rajni Devi

UID- 22BCS15353   Date- 23 Dec ,2024

Section – 620-B

Very Easy

Ques 1:

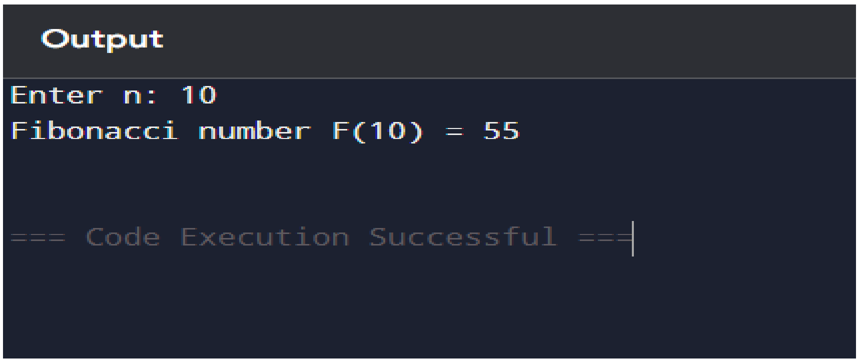
Fibonnacci Series Using Recursion

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1.

Code:

```
#include<iostream>
using namespace std;
int fibonacci(int n) {
   if (n <= 1)
      return n; // Base case: F(0) = 0, F(1) = 1
   return fibonacci(n - 1) + fibonacci(n - 2); // Recursive formula
}
int main() {
   int n;
   cout << "Enter n: ";
   cin >> n;
   cout << "Fibonacci number F(" << n << ") = " << fibonacci(n) << endl;
   return 0;
}
```

Output:



```
Enter n: 10
Fibonacci number F(10) = 55


=== Code Execution Successful ===
```

Ques 2:

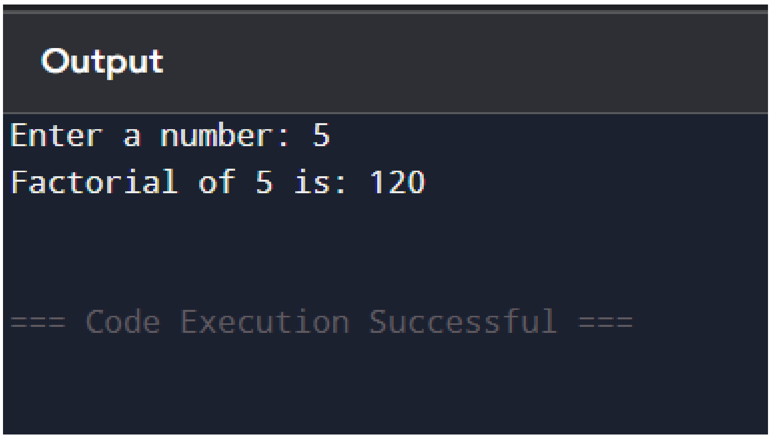
Factorial Of Number Using Recursion

Write a program that returns the value of N! (N factorial) using recursion.
Note that N! =- 1*2*...*N
Also, 0! = 1 and 1! = 1.

Code:

```
#include<iostream>
using namespace std;
int factorial(int n) {
    if (n <= 1)
        return 1; // Base case: 0! = 1 and 1! = 1
    return n * factorial(n - 1); // Recursive case: n! = n * (n-1)!
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    if (n < 0) {
        cout << "Factorial is not defined for negative numbers." << endl;
    } else {
        cout << "Factorial of " << n << " is: " << factorial(n) << endl;
    }
    return 0;
}
```

Output:



```
Enter a number: 5
Factorial of 5 is: 120


=== Code Execution Successful ===
```

Easy

Ques 3:

Reverse Linked List

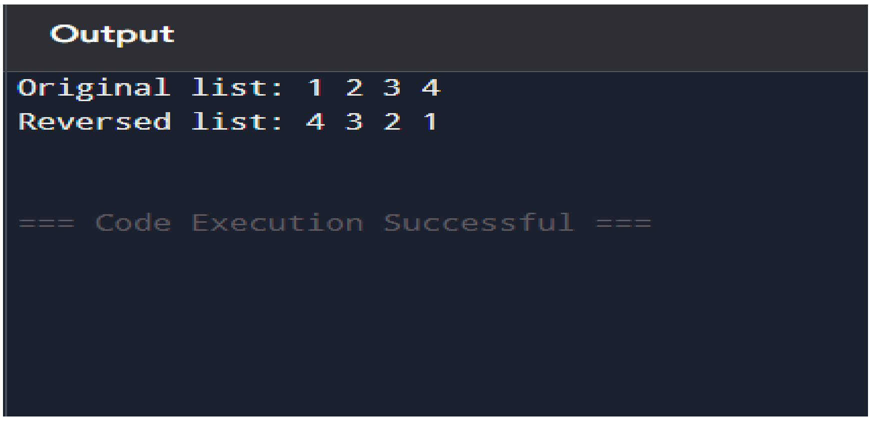
Given the head of a singly linked list, reverse the list, and return the reversed list.

Code:

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
Node* reverseLinkedList(Node* head) {
    Node* prev = nullptr;
    Node* current = head;
    Node* next = nullptr;

    while (current != nullptr) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

```
void push(Node** head, int data) {
   Node* newNode = new Node();
   newNode->data = data;
   newNode->next = *head;
   *head = newNode;
}
int main() {
   Node* head = nullptr;
   push(&head, 4);
   push(&head, 3);
   push(&head, 2);
   push(&head, 1);
   cout << "Original list: ";
   printList(head);
   head = reverseLinkedList(head);
   cout << "Reversed list: ";
   printList(head);
   return 0;
}
```

Output:



```
Original list: 1 2 3 4
Reversed list: 4 3 2 1


=== Code Execution Successful ===
```

<u>Ques 4:</u>

Merge Two Sorted Lists

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

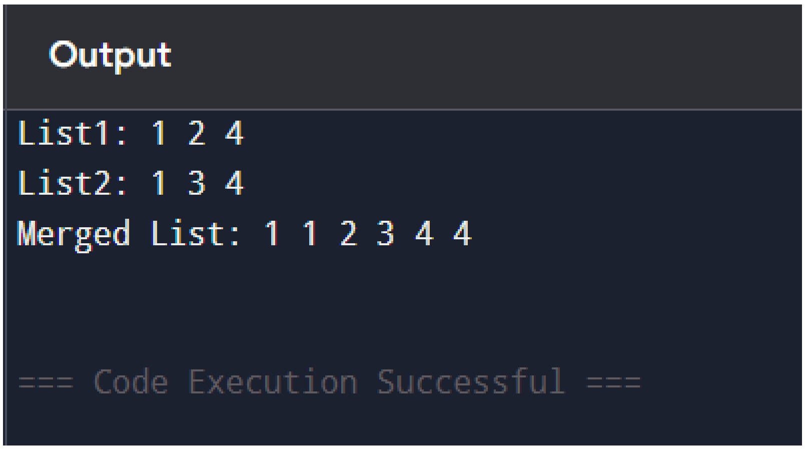
Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

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 list1 is empty, return list2
    if (!list2) return list1; // If list2 is empty, 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) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

ListNode* createList(int arr[], int size) {
    if (size == 0) return nullptr;
    ListNode* head = new ListNode(arr[0]);
    ListNode* current = head;
    for (int i = 1; i < size; i++) {
        current->next = new ListNode(arr[i]);
        current = current->next;
    }
    return head;
}
int main() {
    // Input lists
    int arr1[] = {1, 2, 4};
    int arr2[] = {1, 3, 4};
    ListNode* list1 = createList(arr1, 3);
    ListNode* list2 = createList(arr2, 3);
    cout << "List1: ";
    printList(list1);
    cout << "List2: ";
    printList(list2);
    ListNode* mergedList = mergeTwoLists(list1, list2);
    cout << "Merged List: ";
    printList(mergedList);
    return 0;
}
```

Output:



```
List1: 1 2 4
List2: 1 3 4
Merged List: 1 1 2 3 4 4


=== Code Execution Successful ===
```

Medium

Ques 5:

Add Two Numbers

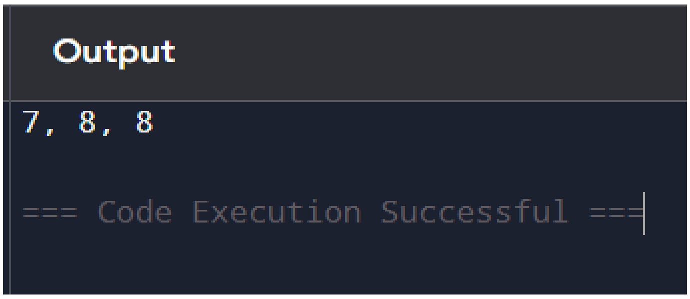
You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Code:

```
#include <iostream>

using namespace std;

int main() {

    const int size = 3;

    int a[size] = {2, 4, 3};

    int b[size] = {5, 4, 5};

    int result[size];

    for (int i = 0; i < size; ++i) {

        result[i] = a[i] + b[i];}

    for (int i = 0; i < size; ++i) {

        cout << result[i];

        if (i != size - 1) { // Add a comma and space except for the last element

            cout << ", ";}}

    return 0;}
```

Output:



```
7, 8, 8

=== Code Execution Successful ===
```

Ques 6:

Minimun non zero product of an Array Elements

You are given a positive integer p. Consider an array nums (1-indexed) that consists of the integers in the inclusive range [1, 2p - 1] in their binary representations. You are allowed to do the following operation any number of times:

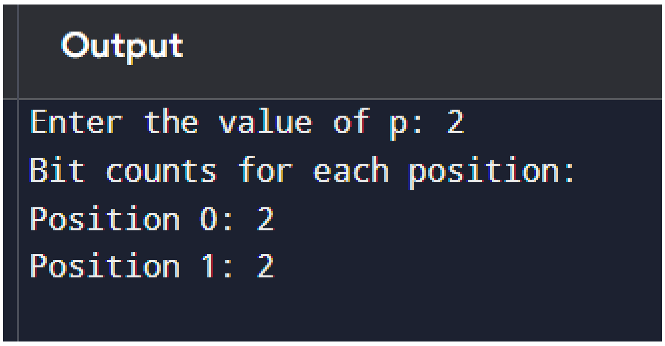
Choose two elements x and y from nums.

Choose a bit in x and swap it with its corresponding bit in y. Corresponding bit refers to the bit that is in the same position in the other integer.

Code:

```
#include <iostream>
using namespace std;
int main() {
  int p;
  cout << "Enter the value of p: ";
  cin >> p;
  int n = (1 << p) - 1; // Largest number in the range [1, 2^p - 1]
  int bitCount[32] = {0}; // Assuming p <= 32
  for (int i = 1; i <= n; ++i) {
    for (int j = 0; j < p; ++j) {
      if (i & (1 << j)) {
        bitCount[j]++;}}}
  cout << "Bit counts for each position:\n";
  for (int i = 0; i < p; ++i) {
    cout << "Position " << i << ": " << bitCount[i] << endl;
  return 0;}
```

Output:



```
Enter the value of p: 2
Bit counts for each position:
Position 0: 2
Position 1: 2
```

Hard

Ques 7:

Permutation Sequence

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

"123"

"132"

"213"

"231"

"312"

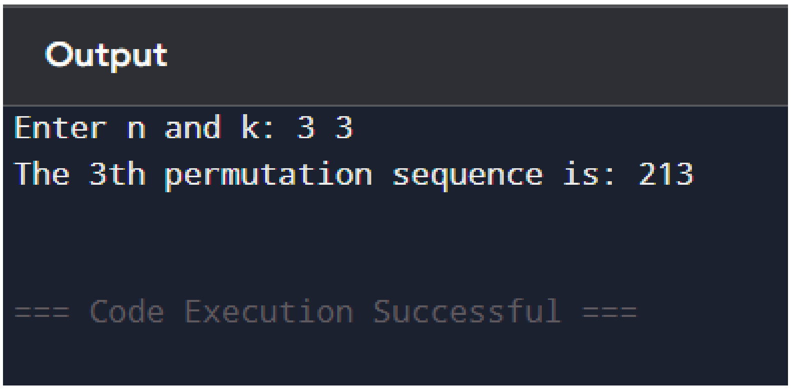
"321"

Given n and k, return the kth permutation sequence.

Code:

```
#include <iostream>

using namespace std;

string getPermutation(int n, int k) {

   int fact = 1;

   bool used[10] = {false}; // Track which numbers are used (1-indexed)

   string result = "";

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

      fact *= i;

   }    k--;

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

      fact /= (n - i); // Factorial for the remaining positions

      int index = k / fact; // Determine which number to pick

      k %= fact;

      int count = 0;

      for (int j = 1; j <= n; ++j) {

         if (!used[j]) {
           if (count == index) {
               result += (j + '0'); // Append the number to result
               used[j] = true;
               break;
           }
           count++;}}}
   return result;
int main() {
   int n, k;
   cout << "Enter n and k: ";
   cin >> n >> k;
   cout << "The " << k << "th permutation sequence is: " << getPermutation(n, k) << endl;
   return 0;
}
```

Output:



```
Enter n and k: 3 3
The 3th permutation sequence is: 213


=== Code Execution Successful ===
```

Ques 8:

Basic Calculator

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

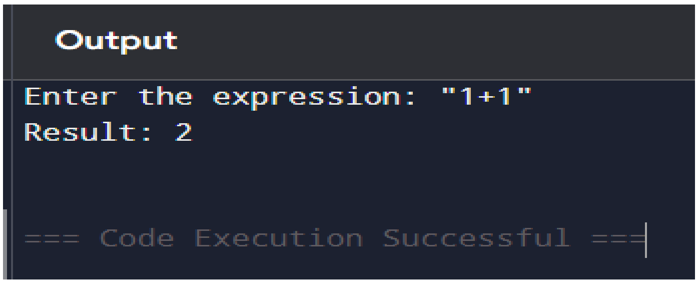
Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

Code:

```
#include <iostream>
#include <string>
```

```
using namespace std;
int calculate(string s) {
    int result = 0, num = 0, sign = 1; // Initialize result, current number, and sign
    for (char c : s) {
        if (isdigit(c)) { // If it's a digit, build the number
            num = num * 10 + (c - '0');
        } else if (c == '+') { // On encountering '+', add the number to result
            result += sign * num;
            num = 0;
            sign = 1; // Reset sign to positive
        } else if (c == '-') { // On encountering '-', subtract the number
            result += sign * num;
            num = 0;
            sign = -1;}}
    result += sign * num; // Add the last number to the result
    return result;}
int main() {
    string s;
    cout << "Enter the expression: ";
    getline(cin, s);
    cout << "Result: " << calculate(s) << endl;
    return 0;
}
```

Output:



```
Enter the expression: "1+1"
Result: 2


=== Code Execution Successful ===
```

Very Hard

Ques 9:

Maximize Number of Nice Divisors

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.

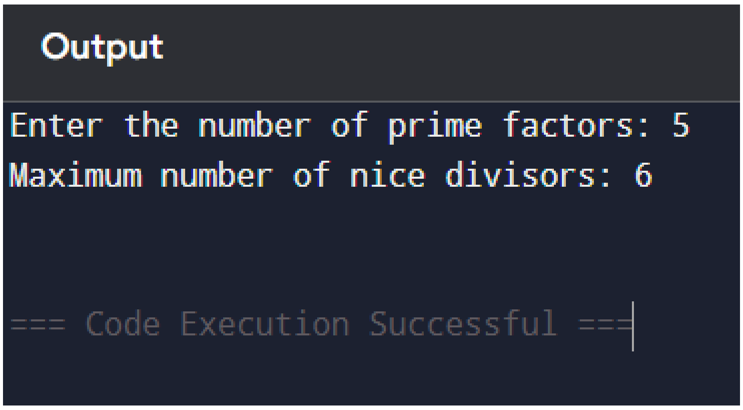
Return the number of nice divisors of n. Since that number can be too large, return it modulo 109 + 7.

Code:

```
#include <iostream>

using namespace std;

const int MOD = 1e9 + 7;

// Function for modular exponentiation

long long modPow(long long base, long long exp, long long mod) {

   long long result = 1;

   while (exp > 0) {

      if (exp % 2 == 1) { // If the current bit is set

         result = (result * base) % mod;

      }

      base = (base * base) % mod;

      exp /= 2;

   }

   return result;

}

int maxNiceDivisors(int primeFactors) {

   if (primeFactors == 1) return 1;

   int groupsOf3 = primeFactors / 3;

   int remainder = primeFactors % 3;
```

```
    if (remainder == 0) {
        return modPow(3, groupsOf3, MOD);
    } else if (remainder == 1) {
        return (modPow(3, groupsOf3 - 1, MOD) * 4) % MOD;
    } else { // remainder == 2
        return (modPow(3, groupsOf3, MOD) * 2) % MOD;
    }
}
int main() {
    int primeFactors;
    cout << "Enter the number of prime factors: ";
    cin >> primeFactors;
    cout << "Maximum number of nice divisors: " << maxNiceDivisors(primeFactors) << endl;
    return 0;
}
```

Output:



```
Enter the number of prime factors: 5
Maximum number of nice divisors: 6


=== Code Execution Successful ===
```

Ques 10:

Q1 There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

Each child must have at least one candy.

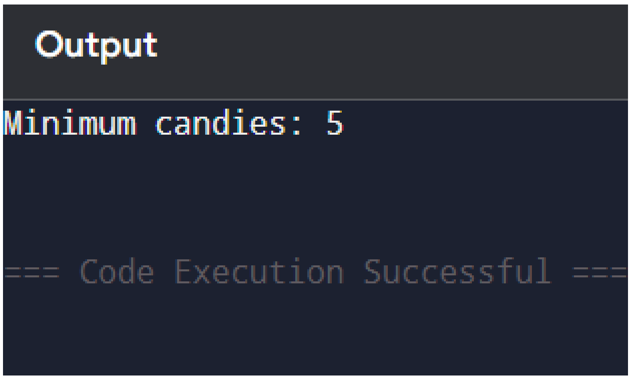
Children with a higher rating get more candies than their neighbors.

Return the minimum number of candies you need to have to distribute the candies to the children.

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int candy(vector<int>& ratings) {
    int n = ratings.size();
    vector<int> candies(n, 1);  // Initialize all candies to 1
    for (int i = 1; i < n; i++) {
        if (ratings[i] > ratings[i - 1]) {
            candies[i] = candies[i - 1] + 1;}}
    for (int i = n - 2; i >= 0; i--) {
        if (ratings[i] > ratings[i + 1]) {
            candies[i] = max(candies[i], candies[i + 1] + 1);}}
    int totalCandies = 0;
    for (int candy : candies) {
        totalCandies += candy;
    }
    return totalCandies;}
int main() {
    vector<int> ratings = {1, 0, 2};
    cout << "Minimum candies: " << candy(ratings) << endl;  // Output: 5
    return 0;}
```

Output:



```
Minimum candies: 5


=== Code Execution Successful ===
```