1. Write a Program to Rotate a matrix by 90 degrees in the clockwise direction in C.

```
// C Program to rotate the array
// By 90 degree in clockwise direction
#include <stdio.h>

void swap(int* a, int* b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main()
{

    int n = 4;
    int arr[4][4] = { { 1, 2, 3, 4 },
                      { 5, 6, 7, 8 },
                      { 9, 10, 11, 12 },
                      { 13, 14, 15, 16 } };

    // Print Orignal Matrix
    printf("Orignal Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

    // Rotate the matrix about the main diagonal
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++)
            swap(&arr[i][j], &arr[j][i]);
    }

    // Rotate the matrix about middle column
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n / 2; j++) {
            swap(&arr[i][j], &arr[i][n - j - 1]);
        }
    }

    // Print the rotated matrix
    printf("Matrix after rotation: \n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr[i][j]);
```

```
        }
        printf("\n");
    }
}
```

2. Write a Program to find the Spiral Traversal of a Matrix in C.

```
// C Program to find Spiral Traversal
// Of a matrix
#include <stdio.h>

int main()
{
    int arr[4][4] = { { 1, 5, 9, 13 },
                      { 2, 6, 10, 14 },
                      { 3, 7, 11, 15 },
                      { 4, 8, 12, 16 } };

    int m = 4, n = 4;
    int i, l = 0, right = m - 1, begin = 0, end = n - 1;

    while (l <= right && begin <= end) {

        // Print the first row
        // from the remaining rows
        for (i = l; i <= right; ++i) {
            printf("%d ", arr[begin][i]);
        }
        begin++;

        // Print the last column
        // from the remaining columns
        for (i = begin; i <= end; ++i) {
            printf("%d ", arr[i][right]);
        }
        right--;

        // Print the last row from
        // the remaining rows
        if (begin <= end) {
            for (i = right; i >= l; --i) {
                printf("%d ", arr[end][i]);
            }
            end--;
        }

        // Print the first column from
        // the remaining columns
```

```
        if (l <= right) {
            for (i = end; i >= begin; --i) {
                printf("%d ", arr[i][l]);
            }
            l++;
        }
    }

    return 0;
}
```

3. Write a program to count the sum of numbers in a string.

```
#include <stdio.h>

int main()
{

    char s[] = "124259";

    int ans = 0;
    // iterate through all the number
    for (int i = 0; s[i] != '\0'; i++) {
        int ele = s[i] - 48;
        if (ele <= 9)
            ans += ele;
    }

    // print sum of the numbers
    printf("%d", ans);

    return 0;
}
```

4. Program to calculate the length of the string.

```
// C Program to calculate
// length of a string
#include <stdio.h>
#include <string.h>

int length(char s[], int i)
{
    if (s[i] == '\0')
        return 0;

    return length(s, i + 1) + 1;
}
```

```
int main()
{
    char s[] = "GeeksforGeeks";

    // Calculating using strlen
    int len = strlen(s);
    printf("length using strlen:%d\n", len);

    // Calculating using iteration
    int i;
    for (i = 0; s[i] != '\0'; i++) {
        continue;
    }
    printf("length using iteration:%d\n", i);

    // Calculating using recursion
    int ans = length(s, 0);
    printf("length using recursion:%d\n", ans);
    return 0;
}
```

5. Write a program to check string is a palindrome.

```
// C implementation to check if a given
// string is palindrome or not
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

bool is_palindrome(char* str, int i, int j)
{
    if (i >= j) {
        return true;
    }
    if (str[i] != str[j]) {
        return false;
    }
    return is_palindrome(str, i + 1, j - 1);
}

void check_palindrome(char* str)
{

    // Start from leftmost and
    // rightmost corners of str
    int h = 0;
    int flag = 0;
```

```
    int l = strlen(str) - 1;

    // Keep comparing characters
    // while they are same
    while (h > l) {
        if (str[l++] != str[h--]) {
            printf("%s is not a palindrome\n", str);
            flag = 1;
            break;
            // will break from here
        }
    }

    if (flag == 0)
        printf("%s is a palindrome\n", str);
}

int main()
{
    char str[] = { "GeekeeG" };
    char str2[] = { "GeeksforGeeks" };

    check_palindrome(str);

    printf("Checking %s using recursive approach\n", str2);
    bool ans = is_palindrome(str2,0,strlen(str2)-1);
    if (ans)
        printf("It is Palindrome\n");
    else
        printf("Not a Palindrome\n");

    return 0;
}
```

6. Write a program to print all permutations of a given string in lexicographically sorted order in C.

```
// C Program to print all permutations of a string in sorted
// order.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// function two compare two characters a and b
int compare(const void* a, const void* b)
{
    return (*(char*)a - *(char*)b);
}
```

```
// function two swap two characters a and b
void swap(char* a, char* b)
{
    char t = *a;
    *a = *b;
    *b = t;
}

// function finds the index of the smallest character
int findCeil(char str[], char first, int l, int h)
{
    int ceilIndex = l;

    for (int i = l + 1; i <= h; i++)
        if (str[i] > first && str[i] < str[ceilIndex])
            ceilIndex = i;

    return ceilIndex;
}

// Print all permutations of str in sorted order
void sortedPermutations(char str[])
{
    int size = strlen(str);

    qsort(str, size, sizeof(str[0]), compare);

    int isFinished = 0;
    while (!isFinished) {
        printf("%s \n", str);

        int i;
        for (i = size - 2; i >= 0; --i)
            if (str[i] < str[i + 1])
                break;

        if (i == -1)
            isFinished = 1;
        else {

            int ceilIndex
                = findCeil(str, str[i], i + 1, size - 1);
            swap(&str[i], &str[ceilIndex]);
            qsort(str + i + 1, size - i - 1, sizeof(str[0]),
                  compare);
        }
    }
```

```
}

int main()
{
    char str[] = "123";
    sortedPermutations(str);
    return 0;
}
```

7. Write a program to calculate the Power of a Number using Recursion in C.

```
// C program  to calculate the Power of a Number using
// Recursion
#include <stdio.h>

int power(int a, int b)
{
    if (b == 0)
        return 1;

    return power(a, b - 1) * a;
}

int main()
{
    int a = 4, b = 5;

    int ans = power(a, b);

    printf("%d", ans);
    return 0;
}
```

8. Write a Code to print the Fibonacci series using recursion.

```
// C Program to illustrate
// Fibonacci Series using Recursion
#include <stdio.h>

int fibonacci(int n)
{
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int fibonacci_iteration(int n)
{
```

```
    if (n <= 1)
        return 1;

    int arr[n + 1];
    arr[0] = 1;
    arr[1] = 1;

    for (int i = 2; i < n + 1; i++)
        arr[i] = arr[i - 1] + arr[i - 2];

    return arr[n];
}

int main()
{
    int n = 9;
    printf("Fibonacci using recursion of %d:%d\n", n,
           fibonacci(n));

    n = 11;
    printf("Fibonacci using iteration of %d:%d", n,
           fibonacci_iteration(n));
    return 0;
}
```

9. Write a Program to find the HCF of two Numbers using Recursion.

```
// C program to find
// GCD of two numbers
#include <stdio.h>

// Recursive function to
// Calculate and return gcd of a and b
int gcd(int a, int b)
{
    // Everything divides 0
    if (a == 0)
        return b;
    if (b == 0)
        return a;

    // base case
    if (a == b)
        return a;

    // a is greater
    if (a > b)
        return gcd(a - b, b);
```

```
    return gcd(a, b - a);
}

int main()
{
    int a = 192, b = 36;
    printf("GCD of %d and %d is %d ", a, b, gcd(a, b));
    return 0;
}
```

10. Write a Program in C to reverse a string using recursion.

```
// C program to reverse
// String using recursion
#include <stdio.h>
#include <string.h>

// Using Iteration for reverse
void reverse_iteration(char* str)
{
    int i = 0;
    int j = strlen(str) - 1;

    for (; i < j; i++, j--) {
        char temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}

// Using recursion for reverse
void reverse(char* str)
{
    if (*str) {
        reverse(str + 1);
        printf("%c", *str);
    }
}

int main()
{
    char a[] = "Geeks for Geeks";
    printf("Orignal string:%s\n", a);

    reverse_iteration(a);
    printf("Reverse the string(iteration):%s\n", a);

    printf("Using recursion for reverse:");
```

```
    reverse(a);

    return 0;
}
```