1. Write a Program to return the nth row of Pascal's triangle.

```
// C program to return the Nth row of pascal's triangle
#include <stdio.h>

// Print the N-th row of the Pascal's Triangle
void generateNthrow(int N)
{
    // nC0 = 1
    int prev = 1;
    printf("%d", prev);

    for (int i = 1; i <= N; i++) {
        // nCr = (nCr-1 * (n - r + 1))/r
        int curr = (prev * (N - i + 1)) / i;
        printf(",%d ", curr);
        prev = curr;
    }
}

int main()
{
    int n = 5;
    generateNthrow(n);
    return 0;
}
```

2. Write a program to reverse an Array.

```
// C Program to reverse
// An array
#include <stdio.h>

void reverse(int* arr, int n)
{
    // Swapping front and back elements.
    for (int i = 0, j = n - 1; i < j; i++, j--) {
        int ele = arr[i];
        arr[i] = arr[j];
        arr[j] = ele;
    }
}

int main()
{

    int arr[] = { 1, 2, 3, 4, 5 };
    // Function Call
```

```
    reverse(arr, 5);

    // reverse array element printing
    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

3. Write a program to check the repeating elements in C.

```
// C Program for
// checking duplicate
// values in a array
#include <stdio.h>

int Sort(int arr[], int size)
{
    for (int i = 0; i < size - 1; i++) {

        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

// find repeating element
void findRepeating(int arr[], int n)
{
    int count = 0;
    for (int i = 0; i < n; i++) {

        int flag = 0;
        while (i < n - 1 && arr[i] == arr[i + 1]) {
            flag = 1;
            i++;
        }
        if (flag)
            printf("%d ", (arr[i - 1]));
    }

    return;
}
```

```
int main()
{
    int arr[] = { 1, 3, 4, 1, 2, 3, 5, 5 };

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

      Sort(arr,n);

      findRepeating(arr,n);



    return 0;
}
```

4. Write a Program to print the Maximum and Minimum elements in an array.

```
// C Program for calculating
// maximum and minimum element
#include <stdio.h>

void find_small_large(int arr[], int n)
{
    int min, max;

    // assign first element as minimum and maximum
    min = arr[0];
    max = arr[0];

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

        // finding smallest here
        if (arr[i] < min)
            min = arr[i]; // finding largest here
        if (arr[i] > max)
            max = arr[i];
    }
    printf("Maximum: %d and Minimum: %d\n", min, max);
}

int main()
{
    int arr[] = { 15, 14, 35, 2, 11, 83 };
    int len = sizeof(arr) / sizeof(arr[0]);

    // Function call
```

```
    find_small_large(arr, len);

    return 0;
}
```

5. Write a Program for the cyclic rotation of an array to k positions.

```
// C program to rotate
// Array by  k elements
#include <stdio.h>

// Print array
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}

// Caculates greatest common divisor
int gcd(int a, int b)
{
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}

// Rotate array
void Rotate(int arr[], int k, int N)
{
    int i, j, a, temp;
    k = k % N;

    int rotate = gcd(k, N);

    for (i = 0; i < rotate; i++) {

        temp = arr[i];
        j = i;
        while (1) {
            a = j + k;
            if (a >= N)
                a = a - N;
            if (a == i)
                break;
            arr[j] = arr[a];
```

```
            j = a;
        }
        arr[j] = temp;
    }
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };

    // Rotating array
    Rotate(arr, 2, 5);

    // Printing array
    printArray(arr, 5);

    return 0;
}
```

6. Write a Program to sort First half in Ascending order and the Second in Descending order.

```
// C Program for Sorting
// First half in Ascending order
// and Second Descending order
#include <stdio.h>

void Sort_asc_desc(int arr[], int n)
{
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    // printing first half in ascending order
    for (int i = 0; i < n / 2; i++)
        printf("%d ", arr[i]);

    // printing second half in descending order
    for (int j = n - 1; j >= n / 2; j--)
        printf("%d ", arr[j]);
```

```
}

int main()
{
    int arr[] = { 11, 23, 42, 16, 83, 73, 59 };
    int N = sizeof(arr) / sizeof(arr[0]);

    Sort_asc_desc(arr, N);

    return 0;
}
```

7. Write a Program to print sums of all subsets in an array.

```
// C Program to print sum of
// all subsets
#include <stdio.h>

// Function to print sum of subset
// Using recursion
void subset_sum(int arr[], int i, int j, int sum)
{
    if (i > j) {
        printf("%d ", sum);
        return;
    }

    subset_sum(arr, i + 1, j, sum + arr[i]);
    subset_sum(arr, i + 1, j, sum);
}

// driver code
int main()
{
    int arr[] = { 1, 2, 3 };
    int n = sizeof(arr) / sizeof(arr[0]);

    // Function calling to print subset sum
    subset_sum(arr, 0, n - 1, 0);
    return 0;
}
```

8. Write a Program to Find if there is any subarray with a sum equal to 0.

```
// C Program to check 0 sum
// subarray possible
#include <stdio.h>
```

```
int main()
{
    // array
    int arr[] = { -2, 2, 1, 1, 8 };
    int n = sizeof(arr) / sizeof(arr[0]);

    int flag = 0, sum;

    // Traversing array to check
    for (int i = 0; i < n; i++) {

        for (int j = i; j < n; j++) {
            sum += arr[j];

            if (sum == 0) {
                flag = 1;
                printf(
                    "True subarray with 0 sum is possible");
                break;
            }
        }
    }

    if (flag == 0)
        printf("No such condition");
}
```

9. Write a C program to Implement Kadane's Algorithm

```
// C program to implement Kadane's Algorithm
#include <limits.h>
#include <stdio.h>

int main()
{
    int a[] = { -2, -3, 4, -1, -2, 1, 5, -3 };
    int n = sizeof(a) / sizeof(a[0]);

    int max_so_far = INT_MIN, max_ending_here = 0,
        start = 0, end = 0, s = 0;

    for (int i = 0; i < n; i++) {
        max_ending_here += a[i];

        if (max_so_far < max_ending_here) {
            max_so_far = max_ending_here;
            start = s;
            end = i;
```

```
        }

        if (max_ending_here < 0) {
            max_ending_here = 0;
            s = i + 1;
        }
    }
    printf("Maximum contiguous sum is %d\n", max_so_far);
    printf("Starting index %d Ending index %d", start, end);

    return 0;
}
```

10. Write a Program to find the transpose of a matrix.

```
#include <stdio.h>

// This function stores transpose of A[][] in B[][]
void transpose(int N, int M, int A[M][N], int B[N][M])
{
    int i, j;
    for (i = 0; i < N; i++)
        for (j = 0; j < M; j++)
            B[i][j] = A[j][i];
}

int main()
{
    int M = 3;
    int N = 4;

    int A[3][4] = { { 1, 1, 1, 1 },
                    { 2, 2, 2, 2 },
                    { 3, 3, 3, 3 } };

    // Note dimensions of B[][]
    int B[N][M], i, j;

    transpose(N, M, A, B);

    printf("Result matrix is \n");
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++)
            printf("%d ", B[i][j]);
        printf("\n");
    }

    return 0;
}
```