1. Write a Program to Replace all 0's with 1's in a Number.

```
// C Program for  Replacing 0 to 1
#include <math.h>
#include <stdio.h>
int main()
{
        int N = 102301;
        int ans = 0;
        int i = 0;
        while (N != 0) {
                // Condition to change value
                if (N % 10 == 0)
                        ans = ans + 1 * pow(10, i);
                else
                        ans = ans + (N % 10) * pow(10, i);

                N = N / 10;
                i++;
        }
        printf("%d", ans);
        return 0;
}
```

2. Write a Program to convert the binary number into a decimal number.

```
// C Program for converting binary to decimal
#include <stdio.h>

int main()
{
   int N = 11011;

   // Initializing base value a to 1
   int a = 1;
   int ans = 0;
   while (N != 0) {
      ans = ans + (N % 10) * a;
      N = N / 10;
      a = a * 2;
   }
   printf("%d", ans);
   return 0;
}
```

3. Program to find the quadrant in which the given coordinates lie

```
// C program to find the quadrant in which the given coordinates lie

#include <stdio.h>
int main()
{
//Fill the code
        int a,b;
        scanf("%d %d",&a,&b);
        if(a > 0 && b > 0)
                printf("Ist Quadrant");
        else if(a < 0 && b > 0)
                printf("IInd Quadrant");
        else if(a < 0 && b < 0)
                printf("IIIrd Quadrant");
        else if(a > 0 && b < 0)
                printf("IVth Quadrant");
        else
                printf("Origin");
        return 0;
}
```

4. Program to find the number of days in a given month

```
// C program to find the number of days in a given month
#include<stdio.h>
int main()
{
   //fill the code
   int year, month;
   scanf("%d %d",&month,&year);
   if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 ||
month == 12)
      printf("Number of days is 31");
   else if((month == 2) && ((year%400==0) || (year%4==0 && year%100!=0)))
   {
      printf("Number of days is 29");
   }
   else if(month == 2)
   {
      printf("Number of days is 28");
   }
   else
      printf("Number of days is 30");
   return 0;
}
```

5. Write a program to form Pascal Triangle using numbers.

```c
// C Program to print Pascal's Triangle
#include <stdio.h>

int main()
{
    int n = 5;


    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n - i; j++) {
            printf("  ");
        }

        int x = 1;

        for (int j = 1; j <= i; j++) {
            printf("%d   ", x);
            x = x * (i - j) / j;
        }
        printf("\n");
    }

    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 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;
}
```

8. Write a program to sort an array using pointers.

```
// C Program to implement sorting using pointers

#include <stdio.h>

// Function to sort the numbers using pointers
void sort(int n, int* ptr)
{
	int i, j;

	// Sort the numbers using pointers
	for (i = 0; i < n; i++) {

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

			if (*(ptr + j) < *(ptr + i)) {

				int temp = *(ptr + i);
				*(ptr + i) = *(ptr + j);
				*(ptr + j) = temp;
			}
		}
	}

	// print the numbers
	for (i = 0; i < n; i++)
		printf("%d ", *(ptr + i));
}

// Driver code
int main()
{
	int n = 5;
	int arr[] = { 13, 22, 7, 12, 4 };

	sort(n, arr);

	return 0;
}
```

9. Write a C Program to Add Two Complex Numbers Using Structures and Functions.

```
// C program to demonstrate addition of complex numbers
#include <stdio.h>
// define a structure for complex number
typedef struct complexNumber {
	int real;
	int img;
} complex;
complex add(complex x, complex y)
{
	// define a new complex number.
	complex add;

	// add similar type together
	add.real = x.real + y.real;
	add.img = x.img + y.img;

	return (add);
}
int main()
{
	// define three complex type numbers
	complex x, y, sum;

	// first complex number
	x.real = 4;
	x.img = 5;

	// second complex number
	y.real = 7;
	y.img = 11;

	// printing both complex numbers
	printf(" x = %d + %di\n", x.real, x.img);
	printf(" y = %d + %di\n", y.real, y.img);

	// call add(a,b) function and
	// pass complex numbers a & b
	// as an parameter.
	sum = add(x, y);

	// print result
	printf("\n sum = %d + %di", sum.real, sum.img);
	return 0;
}
```

10. C Program that will help to remove duplicates from an array.

```
// 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;
}
```