1. Find the largest number among the three numbers.

```
// C Program to find
// Largest of three numbers
#include <stdio.h>

int main()
{
    int a = 1, b = 2, c = 3;

    // condition for a is greatest
    if (a > b && a > c)
        printf("%d", a);

    // condition for b is greatest
    else if (b > a && b > c)
        printf("%d", b);

    // remaining conditions
    // c is greatest
    else
        printf("%d", c);

    return 0;
}
```

2. Write a Program to check whether a number is prime or not.

```
// C Program for Checking value is
// Prime or not
#include <stdbool.h>
#include <stdio.h>

int main() {
    int n = 91;

    int cnt = 0;

    // If number is less than/equal to 1,
    // it is not prime
    if (n <= 1)
        printf("%d is NOT prime\n", n);
    else {

        // Check for divisors from 1 to n
        for (int i = 1; i <= n; i++) {

            // Check how many number is divisible
            // by n
            if (n % i == 0)
                cnt++;
        }

        // If n is divisible by more than 2 numbers
        // then it is not prime
        if (cnt > 2)
            printf("%d is NOT prime\n", n);

        // else it is prime
        else
            printf("%d is prime", n);
    }

    return 0;
}
```

3. Write a C program to calculate Compound Interest.

```
// C program to calculate Compound Interest
#include <stdio.h>

// For using pow function we must
// include math.h
#include <math.h>

// Driver code
int main()
{
    // Principal amount
    double principal = 2300;

    // Annual rate of interest
    double rate = 7;

    // Time
    double time = 4;

    // Calculating compound Interest
    double amount
        = principal * ((pow((1 + rate / 100), time)));
    double CI = amount - principal;

    printf("Compound Interest is : %lf", CI);
    return 0;
}
```

4. Write a Program in C to Swap the values of two variables without using any extra variable

```
// C Program to
// Swap two numbers
// No Extra Space
#include <stdio.h>

int main()
{

    int x = 10;
    int y = 20;

    printf("x: %d , y: %d\n", x, y);

    // x hold 30
    x = x + y;

    // y hold 10
    y = x - y;

    // Now, x hold 20
    x = x - y;

    printf("x: %d , y: %d\n", x, y);

    return 0;
}
```

5. 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;
}
```

6. 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;
}
```

7. Write a Program to check if the year is a leap year or not.

```
// C Program to check
// Year is leap year or not
#include <stdio.h>

// Function Declaration to check leap year
void leap_year(int year)
{
    // If a year is multiple of 400, then leap year
    if (year % 400 == 0)
        printf("%d is a leap year.\n", year);

    // If a year is multiple of 100, then not a leap year
    else if (year % 100 == 0)
        printf("%d is not a leap year.\n", year);

    // If a year is multiple of 4, then leap year
    else if (year % 4 == 0)
        printf("%d is a leap year.\n", year);

    // Not leap year
    else
        printf("%d is not a leap year.\n", year);
}

int main()
{
    leap_year(2000);
    leap_year(2002);
    leap_year(2008);

    return 0;
}
```

8. Write a program to Factorial of a Number.

```
// C Program to calculate
// Factorial of a number
#include <stdio.h>

// Calculating factorial using iteration
void factorial_iteration(int N)
{
    unsigned long long int ans = 1;
    for (int i = 1; i <= N; i++) {
        ans = ans * i;
    }

    printf("Factorial of %d is %lld\n", N, ans);
}

// Calculating factorial using recursion
int factorial(int N)
{
    if (N == 0)
        return 1;

    // Recursive call
    return N * factorial(N - 1);
}

int main()
{
    int n;
    n = 13;
    factorial_iteration(n);

    n = 9;
    printf("Factorial of %d using recursion:%d\n", n,
           factorial(n));

    return 0;
}
```

9. Write a Program to Check if a number is an Armstrong number or not.

```
// C program to check if number
// is Armstrong number or not
#include <stdio.h>

// Function to calculate x raised to the power y
int power(int x, unsigned int y)
{
    if (y == 0)
        return 1;
    if (y % 2 == 0)
        return power(x, y / 2) * power(x, y / 2);

    return x * power(x, y / 2) * power(x, y / 2);
}

// Function to calculate order of the number
int order(int n)
{
    int res = 0;
    while (n) {
        res++;
        n = n / 10;
    }
    return res;
}

// Function to check whether the given number is
// Armstrong number or not
int isArmstrong(int x)
{
    // Calling order function
    int n = order(x);
    int temp = x, sum = 0;
    while (temp) {
        int r = temp % 10;
        sum += power(r, n);
        temp = temp / 10;
    }

    // If satisfies Armstrong condition
    if (sum == x)
        return 1;
    else
        return 0;
```

```
}

// Driver Program
int main()
{
    int x = 120;
    if (isArmstrong(x) == 1)
        printf("True\n");
    else
        printf("False\n");

    x = 1634;
    if (isArmstrong(x) == 1)
        printf("True\n");
    else
        printf("False\n");

    return 0;
}
```

10. Write a program to Find all the roots of a quadratic equation in C.

```
// C program to find roots
// of a quadratic equation
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

// Prints roots of quadratic equation ax*2 + bx + x
void find_roots(int a, int b, int c)
{
    // If a is 0, then equation is not quadratic, but
    // linear
    if (a == 0) {
        printf("Invalid");
        return;
    }

    int d = (b * b) - (4 * a * c);
    double sqrt_val = sqrt(abs(d));

    if (d > 0) {
        printf("Roots are real and different \n");
        printf("%f\n%f", (double)(-b + sqrt_val) / (2 * a),
               (double)(-b - sqrt_val) / (2 * a));
```

```
    }
    else if (d == 0) {
        printf("Roots are real and same \n");
        printf("%f", -(double)b / (2 * a));
    }
    else // d < 0
    {
        printf("Roots are complex \n");
        printf("%f + i%f\n%f - i%f", -(double)b / (2 * a),
               sqrt_val / (2 * a), -(double)b / (2 * a),
               sqrt_val / (2 * a));
    }
}

// Driver code
int main()
{
    int a = 1, b = -16, c = 1;

    // Function call
    find_roots(a, b, c);
    return 0;
}
```