Programs

1.) C Program for the Bisection method

```
#include <stdio.h>
#include <math.h>

// Function to find root of
double func(double x) {
   return x * x * x - x - 2; // Example: f(x) = x^3 - x - 2
}

// Bisection Method
void bisection(double a, double b, double tolerance) {
   double c;
   if (func(a) * func(b) >= 0) {
      printf("Incorrect a and b, function has same signs at both
endpoints.\n");
      return;
   }

   printf("a\t\tb\t\tc\t\tf(c)\n");
   while ((b - a) >= tolerance) {
      c = (a + b) / 2; // Midpoint
      printf("%lf\t%lf\t%lf\t%lf\n", a, b, c, func(c));

      // Check if midpoint is the root
      if (func(c) == 0.0)
         break;

      // Decide which half to take
```

```
        if (func(c) * func(a) < 0)
            b = c;
        else
            a = c;
    }

    printf("\nThe root is approximately: %lf\n", c);
}

int main() {
    double a, b, tolerance;

    // Input for intervals a and b, and tolerance
    printf("Enter the values of a and b:\n");
    scanf("%lf %lf", &a, &b);
    printf("Enter the tolerance value:\n");
    scanf("%lf", &tolerance);

    bisection(a, b, tolerance);

    return 0;
}
```

2.) C Program for the Regula Falsi Method

```
#include <stdio.h>
#include <math.h>

// Function to find the root of
double func(double x) {
    return x * x * x - x - 2;  // Example: f(x) = x^3 - x - 2
}
```

```
// Regula Falsi Method
void regulaFalsi(double a, double b, double tolerance, int maxIterations) {
   double c;
   if (func(a) * func(b) >= 0) {
      printf("Incorrect a and b, function has same signs at both
endpoints.\n");
      return;
   }

   printf("a\t\tb\t\tc\t\tf(c)\n");

   for (int i = 0; i < maxIterations; i++) {
      c = (a * func(b) - b * func(a)) / (func(b) - func(a));  // Formula for
Regula Falsi

      printf("%lf\t%lf\t%lf\t%lf\n", a, b, c, func(c));

      // Check if c is the root
      if (fabs(func(c)) <= tolerance)
         break;

      // Decide which half to take
      if (func(c) * func(a) < 0)
         b = c;
      else
         a = c;
   }

   printf("\nThe root is approximately: %lf\n", c);
}

int main() {
```

```
    double a, b, tolerance;
    int maxIterations;

    // Input for intervals a and b, tolerance, and max iterations
    printf("Enter the values of a and b:\n");
    scanf("%lf %lf", &a, &b);
    printf("Enter the tolerance value:\n");
    scanf("%lf", &tolerance);
    printf("Enter the maximum number of iterations:\n");
    scanf("%d", &maxIterations);

    regulaFalsi(a, b, tolerance, maxIterations);

    return 0;
}
```

3.) C Program for the newton raphson method

```
#include <stdio.h>
#include <math.h>

// Function whose root we are trying to find
double func(double x) {
    return x * x * x - x - 2;  // Example: f(x) = x^3 - x - 2
}

// Derivative of the function
double derivativeFunc(double x) {
    return 3 * x * x - 1;  // Derivative: f'(x) = 3x^2 - 1
}

// Newton-Raphson Method
void newtonRaphson(double x0, double tolerance, int maxIterations) {
```

```
    double x1;
    int iteration = 0;

    printf("Iteration\t x0\t\t f(x0)\t\t f'(x0)\t\t x1\n");

    while (iteration < maxIterations) {
        // Calculate x1 using the Newton-Raphson formula
        x1 = x0 - func(x0) / derivativeFunc(x0);

        printf("%d\t\t %lf\t %lf\t %lf\t %lf\n", iteration + 1, x0, func(x0),
derivativeFunc(x0), x1);

        // Check if the absolute error is within the tolerance
        if (fabs(x1 - x0) < tolerance) {
            printf("\nThe root is approximately: %lf\n", x1);
            return;
        }

        // Update x0 for the next iteration
        x0 = x1;
        iteration++;
    }

    printf("\nMax iterations reached. The root is approximately: %lf\n",
x1);
}

int main() {
    double x0, tolerance;
    int maxIterations;

    // Input for initial guess, tolerance, and maximum number of iterations
    printf("Enter the initial guess (x0):\n");
```

```
    scanf("%lf", &x0);
    printf("Enter the tolerance value:\n");
    scanf("%lf", &tolerance);
    printf("Enter the maximum number of iterations:\n");
    scanf("%d", &maxIterations);

    newtonRaphson(x0, tolerance, maxIterations);

    return 0;
}
```

4.) C Program for the Iteration Method

```
#include <stdio.h>
#include <math.h>

// Function representing g(x), where x = g(x)
double g(double x) {
    return (x * x + 2) / 3;  // Example: g(x) = (x^2 + 2) / 3
}

// Iteration Method (Fixed-Point Iteration)
void iterationMethod(double x0, double tolerance, int maxIterations) {
    double x1;
    int iteration = 0;

    printf("Iteration\t x0\t\t g(x0)\t\t Error\n");

    while (iteration < maxIterations) {
        // Calculate x1 as g(x0)
        x1 = g(x0);

        printf("%d\t\t %lf\t %lf\t %lf\n", iteration + 1, x0, x1, fabs(x1 - x0));
```

```
        // Check if the absolute error is within the tolerance
        if (fabs(x1 - x0) < tolerance) {
            printf("\nThe root is approximately: %lf\n", x1);
            return;
        }

        // Update x0 for the next iteration
        x0 = x1;
        iteration++;
    }

    printf("\nMax iterations reached. The root is approximately: %lf\n",
x1);
}

int main() {
    double x0, tolerance;
    int maxIterations;

    // Input for initial guess, tolerance, and maximum number of iterations
    printf("Enter the initial guess (x0):\n");
    scanf("%lf", &x0);
    printf("Enter the tolerance value:\n");
    scanf("%lf", &tolerance);
    printf("Enter the maximum number of iterations:\n");
    scanf("%d", &maxIterations);

    iterationMethod(x0, tolerance, maxIterations);

    return 0;
}
```

5.) C program to find absolute error , relative error and percentage error.

```
#include <stdio.h>
#include <math.h>

// Function to calculate absolute error
double absoluteError(double actual, double measured) {
   return fabs(actual - measured);
}

// Function to calculate relative error
double relativeError(double actual, double measured) {
   return fabs((actual - measured) / actual);
}

// Function to calculate percentage error
double percentageError(double actual, double measured) {
   return relativeError(actual, measured) * 100;
}

int main() {
   double actualValue, measuredValue;

   // Input for actual and measured values
   printf("Enter the actual value: ");
   scanf("%lf", &actualValue);

   printf("Enter the measured value: ");
   scanf("%lf", &measuredValue);

   // Calculate errors
   double absError = absoluteError(actualValue, measuredValue);
   double relError = relativeError(actualValue, measuredValue);
```

```
    double percError = percentageError(actualValue, measuredValue);

    // Display the results
    printf("\nAbsolute Error: %lf", absError);
    printf("\nRelative Error: %lf", relError);
    printf("\nPercentage Error: %lf%%\n", percError);

    return 0;
}
```