Q1- Write a C program to find roots of a quadratic equation.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

#include <stdio.h> #include <math.h> void main()

{

float a,b,c,d,r1,r2; printf("Enter coeff. a, b, c: ");

scanf("%f %f %f",&a,&b,&c); d=b\*b-(4\*a\*c);

printf("Equation is: %fx^2 + %fx +%f\n",a,b,c); if (d==0) {

printf("Since discriminant is equal to 0\n"); r1=-b/(2\*a);

printf("Equation has equal roots: %f",r1);

}

else if (d>0) {

printf("Since discriminant is greater than 0\n"); r1=(-b+sqrt(d))/(2\*a);

r2=(-b-sqrt(d))/(2\*a);

printf("Equation has 2 distinct roots: %f and %f",r1,r2);

}

else {

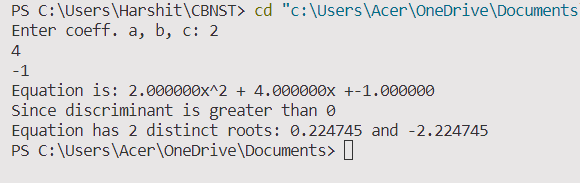
printf("Since discriminant is less than 0\n"); r1=-b/(2\*a);

r2=sqrt(-d)/(2\*a);

printf("Equation has 2 imaginary roots: %f + %fi and %f - %fi",r1,r2,r1,r2);

} }

OUTPUT



Q2- Write a C program to implement Bisection method to find approximate root.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

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

#define f(x) pow(x,3)-2\*x-5 void main()

{

float a,b,x,e,f0,f1,f2; int iter=1;

printf("Enter the value of a: "); scanf("%f",&a);

printf("Enter the value of b: "); scanf("%f",&b);

printf("Enter tolerable error: "); scanf("%f",&e);

f0=f(a);

f1=f(b);

if (f0\*f1>0)

{

printf("Initial values are incorrect"); return;

}

printf("Iterations\tIntermediate roor\n"); do

{

x=(a+b)/2; f2=f(x);

printf("%d\t\t%f\n",iter,x);

if(f0\*f2<0)

{

b=x; f1=f2;

}

else

{

a=x; f0=f2;

}

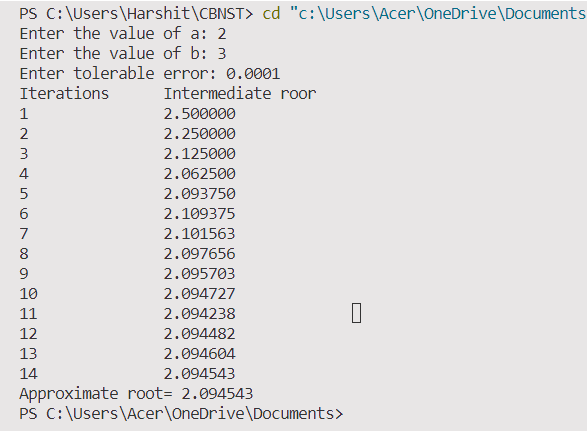
iter+=1;

}

while(fabs(f2)>e); printf("Approximate root= %f",x);

}

OUTPUT



Q3- Write a C program to implement Regular Falsi method to find approximate root.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

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

#define f(x) pow(x,3)-2\*x-5 void main()

{

float a,b,c,fa,fb,fc,e; int iter=1;

printf("Enter the value of a: "); scanf("%f",&a);

printf("Enter the value of b: "); scanf("%f",&b);

printf("Enter tolerable error: "); scanf("%f",&e);

fa=f(a);

fb=f(b);

if (fa\*fb>0)

{

printf("Initial values are incorrect"); return;

}

printf("Iterations\tIntermediate roor\n"); do

{

c=((a\*fb)-(b\*fa))/(fb-fa); fc=f(c); printf("%d\t\t%f\n",iter,c);

if(fa\*fc<0)

{

b=c; fb=fc;

}

else

{

a=c; fa=fc;

}

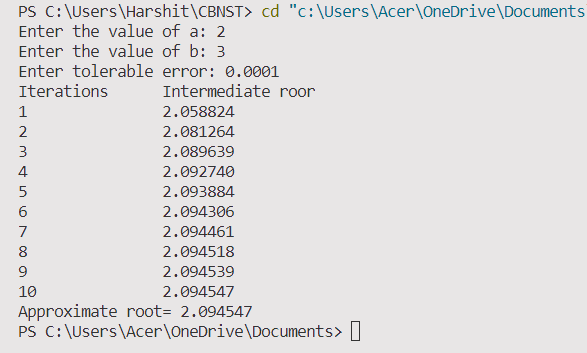
iter+=1;

}

while(fabs(fc)>e); printf("Approximate root= %f",c);

}

OUTPUT



Q4- Write a C program to implement Newton Raphson Method to find approximate root.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

#include <stdio.h> #include <math.h> #define f(x) x\*x\*x-5\*x-3 #define g(x) 3\*x\*x-5 void main()

{

float x0,x1,f0,f1,g0,e,a,b; int step=1,N;

printf("Enter the value of a: "); scanf("%f",&a);

printf("Enter the value of b: "); scanf("%f",&b);

printf("Enter tolerable error: "); scanf("%f",&e);

if (fabs(f(a))>fabs(f(b))) x0=b;

else

x0=a;

if (f(a)\*f(b)>0)

{

printf("Initial values are incorrect"); return;

}

printf("Step\t\txn\t\tf(xn)\t\txn+1\t\tf(xn+1)\n"); do

{

g0=g(x0);

f0=f(x0); if(g0==0.0)

{

printf("Mathematical error"); return;

}

x1=x0-f0/g0; printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,f0,x1,f1); x0=x1;

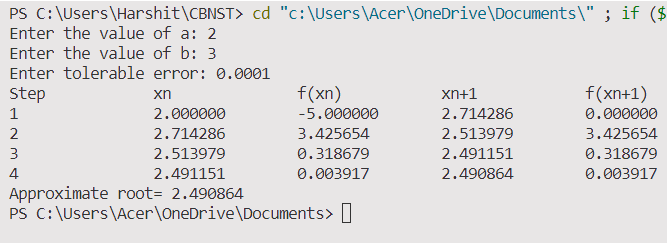
step+=1; f1=f(x1);

}

while(fabs(f1)>e); printf("Approximate root= %f",x1);

}

OUTPUT



Q5- Write a C program to implement Secant method to find approximate root.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/ #include<stdio.h> #include<math.h>

#define f(x) cos(x)+2\*sin(x)+x\*x

void main()

{

float x0, x1, x2, f0, f1, f2, e; int step = 1, N;

printf("Enter 1st initial guess: "); scanf("%f", &x0);

printf("Enter 2nd initial guess: "); scanf("%f", &x1);

printf("Enter tolerable error: "); scanf("%f", &e);

printf("Enter maximum iteration: "); scanf("%d", &N); printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n"); do

{

f0 = f(x0);

f1 = f(x1); if(f0 == f1)

{

printf("Mathematical Error.");

return;

}

x2 = x1 - (x1 - x0) \* f1/(f1-f0); f2 = f(x2);

printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,x1,x2, f2);

x0 = x1; f0 = f1; x1 = x2; f1 = f2;

step = step + 1; if(step > N)

{

printf("Not Convergent."); return;

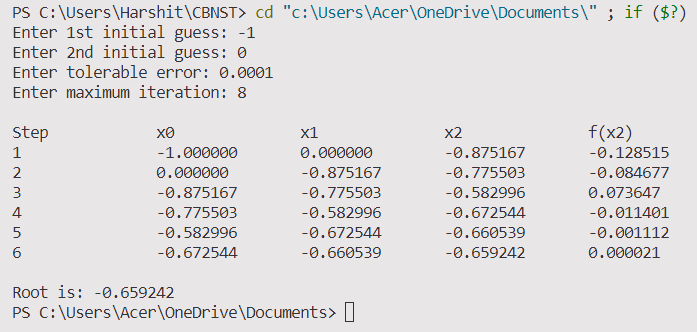
}

}

while(fabs(f2)>e); printf("\nRoot is: %f", x2);

}

OUTPUT



Q6- Write a C program to implement Iteration method to find real roots of non linear expressions.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

double eq(double x)

{

    return x \* x \* x + x \* x - 1;

}

double feq(double x)

{

    return 1 / (sqrt(1 + x));

}

double dfeq(double x)

{

    return 1 / 2.0 \* (1, pow((1 + x \* x), 3 / 2.0));

}

void iterative()

{

    double x0, x1, x2, x3;

    int i;

    while (1)

    {

        printf("Enter first root: ");

        scanf("%lf", &x1);

        printf("Enter second root: ");

        scanf("%lf", &x2);

        printf("%.4lf\n", eq(x1));

        printf("%.4lf\n", eq(x2));

        if (eq(x1) \* eq(x2) < 0)

            break;

        else

            printf("Enter again \n");

    }

    x0 = (x1 + x2) / 2.0;

    if (fabs(dfeq(x0) < 1))

    {

        printf("Iteration can be applied\n");

    }

    else

    {

        printf("Iteration cannot be applied");

    }

    for (i = 0; i < 5; i++)

    {

        x3 = feq(x0);

        printf("The value at iteration %d = %.4lf\n", i + 1, x3);

        x0 = x3;

    }

    printf("The root is %.4lf", x3);

}

int main()

{

    iterative();

    return 0;

}

OUTPUT

A screenshot of a computer program

Description automatically generated

Q7- Write a C program to implement Gauss Elimination method to find real roots of non linear expressions.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define max 10

int main()

{

    float a[max][max], x[max], ratio;

    int i, j, k, n;

    printf("Enter number of unknowns: ");

    scanf("%d", &n);

    printf("\nEnter elements of Augmented Matrix:\n");

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j <= n + 1; j++)

        {

            printf("a[%d][%d] = ", i, j);

            scanf("%f", &a[i][j]);

        }

    }

    printf("\nAugmented Matrix:\n");

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j <= n + 1; j++)

        {

            printf("%0.2f\t", a[i][j]);

        }

        printf("\n");

    }

    for (i = 1; i <= n - 1; i++)

    {

        if (a[i][i] == 0.0)

        {

            printf("Mathematical Error!");

            exit(0);

        }

        for (j = i + 1; j <= n; j++)

        {

            ratio = a[j][i] / a[i][i];

            for (k = 1; k <= n + 1; k++)

            {

                a[j][k] = a[j][k] - ratio \* a[i][k];

            }

        }

    }

    printf("\nUpper Triangular Matrix:\n");

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j <= n + 1; j++)

        {

            printf("%0.2f\t", a[i][j]);

        }

        printf("\n");

    }

    x[n] = a[n][n + 1] / a[n][n];

    for (i = n - 1; i >= 1; i--)

    {

        x[i] = a[i][n + 1];

        for (j = i + 1; j <= n; j++)

        {

            x[i] = x[i] - a[i][j] \* x[j];

        }

        x[i] = x[i] / a[i][i];

    }

    printf("\nSolution:\n");

    for (i = 1; i <= n; i++)

    {

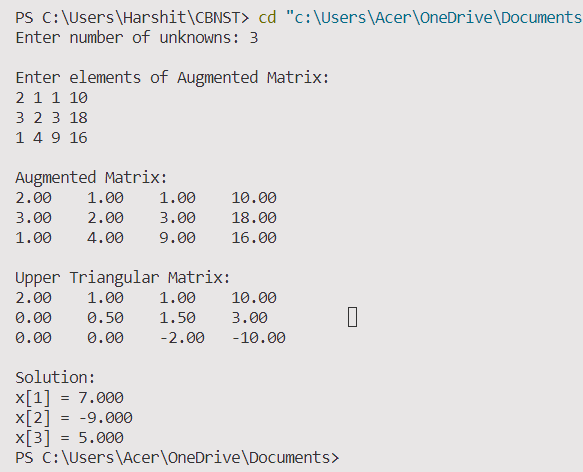
        printf("x[%d] = %0.3f\n", i, x[i]);

    }

    return 0;

}

OUTPUT



Q8- Write a C program to implement Gauss Jordan method to find real roots of non linear expressions.

/\*Name- Harshit Dutt Tyagi Section- G

Roll no.- 22

\*/

#include <stdio.h>

int n;

void convertToDiagnolMatrix(float a[][n + 1], int n)

{

    int i, j, x, y, k;

    float ratio;

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n; j++)

        {

            if (i != j)

            {

                ratio = a[j][i] / a[i][i];

                for (k = 0; k < n + 1; k++)

                    a[j][k] = a[j][k] - (ratio \* a[i][k]);

            }

        }

    }

    printf(" Diagnol Matrix is: \n");

    for (i = 0; i < n; i++)

    {

        for (j = 0; j < n + 1; j++)

        {

            printf("%f ", a[i][j]);

        }

        printf("\n");

    }

}

void directdivide(float a[][n + 1], float value[], int n)

{

    for (int i = 0; i < n; i++)

    {

        value[i] = a[i][n] / a[i][i];

    }

}

void print(float value[], int n)

{

    int i;

    printf("Values of unknowns are:\n");

    for (i = 0; i < n; i++)

        printf("Value[%d]=%f\n", i, value[i]);

}

int main()

{

    int i, j, k, x, y;

    float sum, ratio;

    printf("Enter no of Unknowns\n");

    scanf("%d", &n);

    float a[n][n + 1], value[n];

    printf("Enter the Augmented Matrix\n");

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n + 1; j++)

            scanf("%f", &a[i][j]);

    }

    convertToDiagnolMatrix(a, n);

    directdivide(a, value, n);

    print(value, n);

    return 0;

}

OUTPUT

A screenshot of a computer

Description automatically generated