**Lab-1**

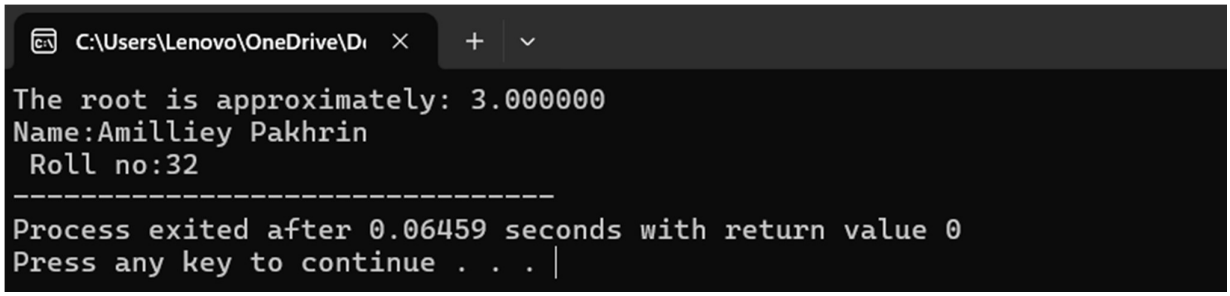
1. **Write a program to calculate the root of 4x3-2x+6 using bisection method.**

```
#include <stdio.h>
#include <math.h>
double f(double x)
{
   return 4 * x * x * x - 2 * x + 6;
}
double bisection(double a, double b, double tol)
{
   if (f(a) * f(b) >= 0)
   {
      printf("The bisection method cannot be applied. f(a) and f(b) must have opposite signs.\n");
      return -1;
   }
   double c = a;
   while ((b - a) / 2.0 > tol)
   {
      c = (a + b) / 2.0;
      if (f(c) == 0.0)
         break;

      if (f(c) * f(a) < 0)
         b = c;
      else
         a = c;
   }
   return c;
}
int main()
{
   double a = -2, b = 2;
   double tolerance = 1e-5;

   double root = bisection(a, b, tolerance);

   if (root != -1)
      printf("The root is approximately: %lf\n", root);
   printf("Name:Amilliey Pakhrin \n Roll no:32");

   return 0;
```

```
C:\Users\Lenovo\OneDrive\D
The root is approximately: -1.289627
Name:Amilliey Pakhrin
 Roll no:32
--------------------------------
Process exited after 0.06841 seconds with return value 0
Press any key to continue . . .
```

**2. Write a program to calculate the root of x2-5x+6 using false position method.**

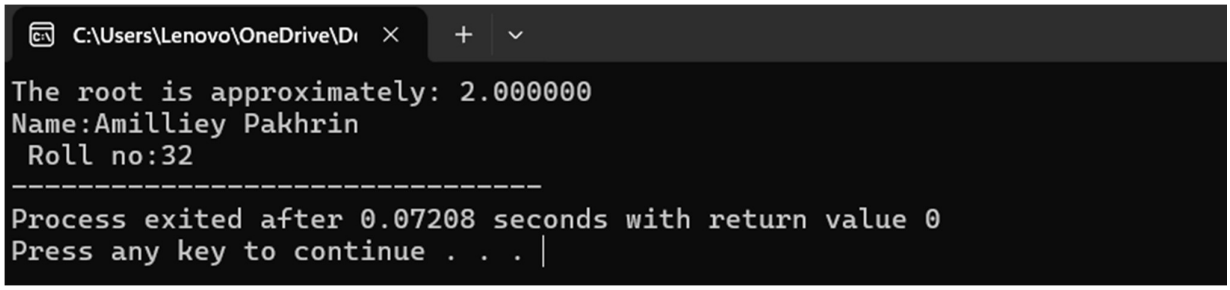
```
#include <stdio.h>
#include <math.h>

double f(double x)
{
   return x * x - 5 * x + 6;
}
double false_position(double a, double b, double tol)
{
   double c = a;

   if (f(a) * f(b) > 0)
   {
      printf("The False Position method cannot be applied. f(a) and f(b) must have opposite signs.\n");
      return -1;
   }
   while ((b - a) / 2.0 > tol)
   {
      c = (a * f(b) - b * f(a)) / (f(b) - f(a));

      if (f(c) == 0.0)
      {
         break;
      }

      if (f(c) * f(a) < 0)
         b = c;
      else
         a = c;
   }
   return c;
}
int main()
{
   double a = 1, b = 3;
   double tolerance = 1e-5;

   double root = false_position(a, b, tolerance);

   if (root != -1)
      printf("The root is approximately: %lf\n", root);
   printf("Name:Amilliey Pakhrin\n Roll no:32");

   return 0;
}
```




3. **Write a program to calculate the root of x3-3x-2 using Newton-Raphson method.**

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

double f(double x)
{
   return x * x * x - 3 * x - 2;
}

double f_prime(double x)
{
   return 3 * x * x - 3;
}

double newton_raphson(double x0, double tol)
{
   double x1;

   while (1)
   {

      x1 = x0 - f(x0) / f_prime(x0);

      if (fabs(x1 - x0) < tol)
      {
         break;
      }

      x0 = x1;
   }

   return x1;
}

int main()
{
   double x0 = 2;
   double tolerance = 1e-5
```

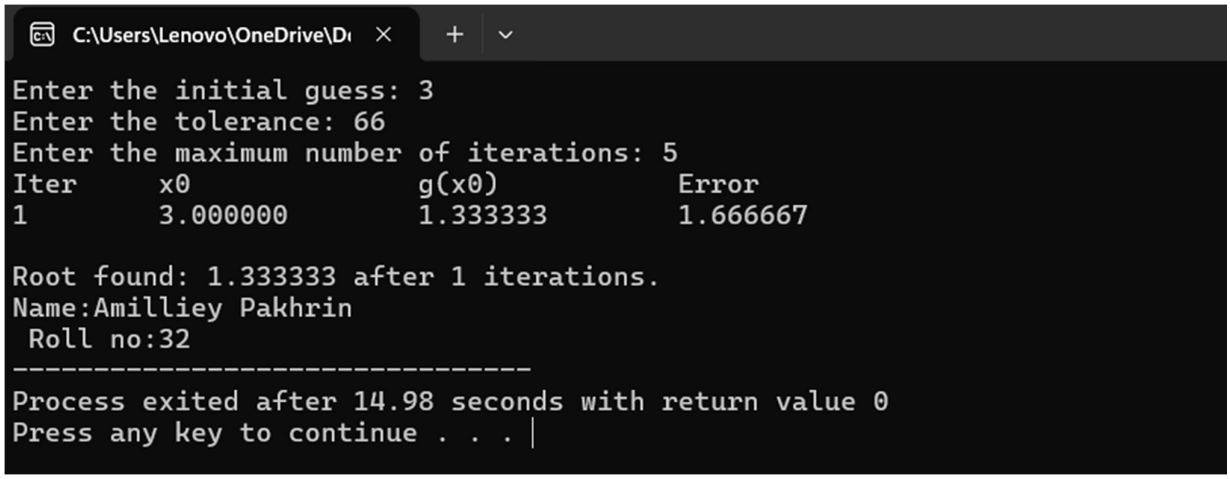
```
    double root = newton_raphson(x0, tolerance);

    printf("The root is approximately: %lf\n", root);
    printf("Name:Amilliey Pakhrin\n Roll no:32");

    return 0;
}
```




4. **Write a program to calculate the root of x2-x-1 using fixed point method. Choose appropriate form of g(x) yourself.**

```
#include <stdio.h>
#include <math.h>
double g(double x)
{
  return 1 + (1 / x);
}
void fixedPoint(double initialGuess, double tolerance, int maxIterations)
{
  double x0 = initialGuess;
  double x1;
  int iteration = 0;

  printf("Iter\t x0\t\t g(x0)\t\t Error\n");
  do
  {
    x1 = g(x0);
    printf("%d\t %lf\t %lf\t %lf\n", iteration + 1, x0, x1, fabs(x1 - x0));
```

```
        if (fabs(x1 - x0) < tolerance)
        {
            printf("\nRoot found: %lf after %d iterations.\n", x1, iteration + 1);
            return;
        }
        x0 = x1;
        iteration++;
    } while (iteration < maxIterations);
    printf("\nMaximum iterations reached. Approximate root: %lf\n", x1);
}
int main()
{
    double initialGuess, tolerance;
    int maxIterations;
    printf("Enter the initial guess: ");
    scanf("%lf", &initialGuess);
    printf("Enter the tolerance: ");
    scanf("%lf", &tolerance);
    printf("Enter the maximum number of iterations: ");
    scanf("%d", &maxIterations);
    fixedPoint(initialGuess, tolerance, maxIterations);
    printf("Name:Amilliey Pakhrin \n Roll no:32");

    return 0;
}
```




**Lab-2**

**5.Write a program to read a set of data points from user and compute interpolation value at specified point using Lagrange interpolation.**

```
#include <stdio.h>
double lagrangeInterpolation(int n, double x[], double y[], double xp)
{
   double result = 0.0;

   for (int i = 0; i < n; i++)
   {
      double term = y[i];

      for (int j = 0; j < n; j++)
      {
         if (i != j)
         {
            term = term * (xp - x[j]) / (x[i] - x[j]);
         }
      }

      result += term;
   }

   return result;
}

int main()
{
   int n;
   printf("Enter the number of data points: ");
   scanf("%d", &n);
```

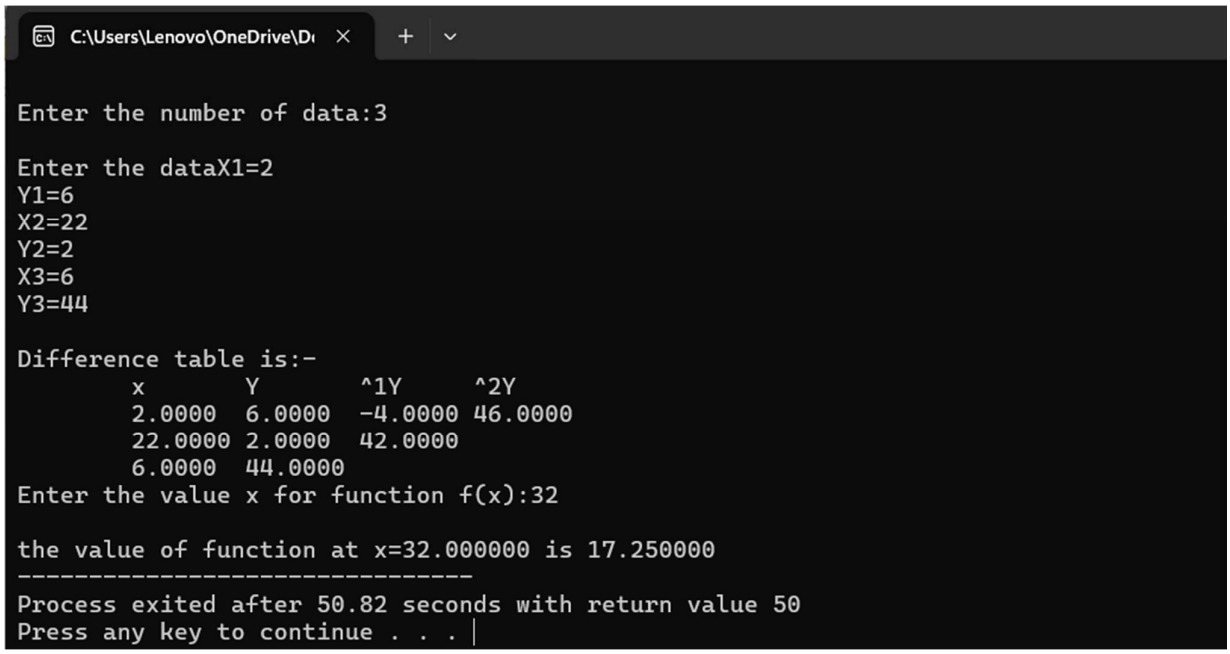
```
    double x[n], y[n];

    printf("Enter the data points (x and y values):\n");
    for (int i = 0; i < n; i++)
    {
        printf("x[%d] = ", i);
        scanf("%lf", &x[i]);
        printf("y[%d] = ", i);
        scanf("%lf", &y[i]);
    }

    double xp;
    printf("Enter the value of x at which to interpolate: ");
    scanf("%lf", &xp);

    double yp = lagrangeInterpolation(n, x, y, xp);

    printf("The interpolated value at x = %lf is y = %lf\n", xp, yp);
    printf("Name:Amilliey Pakhrin \n roll no.32");

    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\Lenovo\OneDrive\Documents\4th sem project\Numerical Methods\codes>  & 'c:\Users\Lenovo\.vscode\
dapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-xtdzjqhr.z0x' '--stdout=Microsoft-MIEn
ror-gycrdeel.yby' '--pid=Microsoft-MIEngine-Pid-xxsmcwmj.sl4' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--in
Enter the number of data points: 5
Enter the data points (x and y values):
x[0] = 21
x[0] = 21
y[0] = 6
x[1] = 34
y[1] = 25
x[2] = 20
y[2] = 66
x[3] = 40
y[3] = 27
x[4] = 33
y[4] = 23
Enter the value of x at which to interpolate: 20
The interpolated value at x = 20.000000 is y = 66.000000
Name:Amilliey Pakhrin
 roll no.32
```

**6. Write a program to read a set of data points from user and compute interpolation value at specified point using Newton interpolation.**

```
#include<stdio.h>
#include<math.h>
int fact(int);
void main(){
float arr[10][11],x,h,p,y,px=1;
int i,j,n,ch=30;
printf("\nEnter the number of data:");
scanf("%d",&n);
printf("\nEnter the data");
for(i=0;i<n;i++){
   printf("X%d=",i+1);
   scanf("%f",&arr[i][0]);
   printf("Y%d=",i+1);
   scanf("%f",&arr[i][1]);
}
//Forming difference table.
for(j=2;j<=n;j++)
for(i=0;i<n-1;i++)
arr[i][j]=arr[i+1][j-1]-arr[i][j-1];
//Printing table
printf("\nDifference table is:-");
printf("\n\tx\tY");
for(i=0;i<=n-2;i++)
   printf("\t%c^%dY",ch,i+1);
for(i=0;i<n;i++){
   printf("\n");
for(j=0;j<n+1-i;j++){
   printf("\t%.4f",arr[i][j]);
   }
}
//Take the value of x for f(x)
printf("\nEnter the value x for function f(x):");
scanf("%f",&x);
//Calculate the value of f(x) for x
h=arr[1][0]-arr[0][0];
p=(x-arr[0][0])/h;
y=arr[0][1];
for(i=1;i<n;i++){
    px=px*(p-(i-1));
   y=y+(arr[0][i+1]*px)/fact(i);
}
printf("\nthe value of function at x=%f is %f",x,y);
 printf("Name:Amilliey Pakhrin \n roll no.32");
}
```

```
int fact(int n){
    int i,f=1;
   for(i=1;i<=n;i++)
   f=f*i;
   return f;
}
```




**7. Write a program to read a set of data points from user and fit the line Y = A + BX through the points by the method of least squares.**

```
#include <stdio.h>

// Function to perform linear regression

void linearRegression(float x[], float y[], int n, float *slope, float *intercept) {

   float sumX = 0.0, sumY = 0.0, sumXY = 0.0, sumX2 = 0.0;

   for (int i = 0; i < n; i++) {

      sumX += x[i];

      sumY += y[i];

      sumXY += x[i] * y[i];

      sumX2 += x[i] * x[i];

   }

   float numerator = n * sumXY - sumX * sumY;

   float denominator = n * sumX2 - sumX * sumX;
```

```
    *slope = numerator / denominator;
    *intercept = (sumY - (*slope) * sumX) / n;
}
int main() {
    int n;
    printf("Enter the number of data points: ");
    scanf("%d", &n);

    float x[n], y[n];
    printf("Enter data points in the format 'x y':\n");
    for (int i = 0; i < n; i++) {
        scanf("%f %f", &x[i], &y[i]);
    }

    float slope, intercept;
    linearRegression(x, y, n, &slope, &intercept);

    printf("Fitted straight line equation: y = %.4fx + %.4f\n", slope, intercept);
     printf("Name:Amilliey Pakhrin \n roll no.32");

    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\Lenovo\OneDrive\Documents\4th sem project\Numerical Methods\codes>  & 'c:\Users\
dapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-cnhzlmed.l2y' '--stdout
ror-ip4emw3b.p5l' '--pid=Microsoft-MIEngine-Pid-o1sogm04.tfl' '--dbgExe=C:\msys64\ucrt64\bin
Enter the number of data points: 2
Enter data points in the format 'x y':
33 23
11 21
Fitted straight line equation: y = 0.0909x + 20.0000
Name:Amilliey Pakhrin
 roll no.32
```

**Lab-3**

**8. Write a program to integrate a given function using trapezoidal rule.**

```
#include<stdio.h>
#define f(x) pow(x,3)+3*x
float findValueAt(float x){
   return f(x);
}
int main(){
   int n;
   float i,a,b,sum=0,h;
   //Input
   printf("Enter Value of a and b\n");
   scanf("%f%f",&a,&b);
   printf("Enter no. of Intervals\n");
   scanf("%d",&n);
   h=(b-a)/n;
   sum = findValueAt(a) +findValueAt(b);
   for(i=a+h;i<b;i=i+h)
     sum = sum + 2*findValueAt(i);
   sum = (h * sum)/2;
   //Print the Output
   printf("\nValue of The integral  = %f",sum);
 printf("Name:Amilliey Pakhrin \n roll no.32");
}
```

```
Enter the lower limit of integration (a): 5
Enter the upper limit of integration (b): 32
Enter the number of intervals (n): 3

The integral of the function from 5.00 to 32.00 is approximately: 11245.500000
Name:Amilliey Pakhrin
Roll no:32
```

**9. Write a program to integrate a given function using Simpsons 1/3 rule.**

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

```
#define f(x) pow(x,3)+3*x
float findValueAt(float x){
   return f(x);
}
int main(){
   int n;
   float i,a,b,sum=0,h;
   //The initial Position (0) is treated as Even position
   int position_of_term=1;
   //Input
   printf("Enter Value of a and b\n");
   scanf("%f%f",&a,&b);
   printf("Enter no. of Intervals\n");
   scanf("%d",&n);

   h=(b-a)/n;
   sum = findValueAt(a) +findValueAt(b);

   for(i=a+h;i<b;i=i+h)
   {
     if(position_of_term %2 ==0)
       sum = sum + 2*findValueAt(i);
     else
       sum = sum + 4*findValueAt(i);
     position_of_term++;
   }
   sum = (h * sum)/3;
   //Print the Output
   printf("\nValue of The integral  = %f",sum);
            printf("Name:Amilliey Pakhrin \n roll no.32");
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensions\ms-vscode
er.exe' '--stdin=Microsoft-MIEngine-In-w1og0ei0.jxd' '--stdout=Microsoft-MIEngine-Out-0meqnxik.d
soft-MIEngine-Pid-hgcwdici.dpn' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the lower limit of integration (a): 34
Enter the upper limit of integration (b): 21
Enter the number of intervals (n, must be even): 4

The integral of the function from 34.00 to 21.00 is approximately: -10014.333333
Name:Amilliey Pakhrin
Roll no:32
```

**Lab-3**

**10. Write a program to solve system of nonlinear equations using Gauss-Elimination method**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX 10
void gaussElimination(int n, double a[MAX][MAX], double b[MAX], double x[MAX])
{
   int i, j, k;
   double factor;
   for (k = 0; k < n - 1; k++)
   {
      for (i = k + 1; i < n; i++)
      {
         if (a[k][k] == 0)
         {
            printf("Mathematical Error: Division by zero.\n");
            exit(1);
         }
         factor = a[i][k] / a[k][k];
         for (j = k; j < n; j++)
         {
            a[i][j] -= factor * a[k][j];
         }
```

```
            b[i] -= factor * b[k];
        }  }
    for (i = n - 1; i >= 0; i--)
    {
        x[i] = b[i];
        for (j = i + 1; j < n; j++)
        {
            x[i] -= a[i][j] * x[j];
        }
        if (a[i][i] == 0)
        {
            printf("Mathematical Error: Division by zero.\n");
            exit(1);
        }
        x[i] /= a[i][i];
    }
}
int main()
{
    int n, i, j;
    double a[MAX][MAX], b[MAX], x[MAX];
    printf("Enter the number of equations: ");
    scanf("%d", &n);
    if (n > MAX)
    {
        printf("Maximum number of equations is %d.\n", MAX);
        return 1;
    }
    printf("Enter the coefficients of the equations:\n");
    for (i = 0; i < n; i++)
    {
```

```
        for (j = 0; j < n; j++)
        {
            printf("a[%d][%d] = ", i + 1, j + 1);
            scanf("%lf", &a[i][j]);
        }
    }
    printf("Enter the constant terms:\n");
    for (i = 0; i < n; i++)
    {
        printf("b[%d] = ", i + 1);
        scanf("%lf", &b[i]);
    }
    gaussElimination(n, a, b, x);
    printf("\nSolution:\n");
    for (i = 0; i < n; i++)
    {
        printf("x[%d] = %.6lf\n", i + 1, x[i]);
    }
    printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensions\ms-vscode.cppt
er.exe' '--stdin=Microsoft-MIEngine-In-k3bkjczl.1im' '--stdout=Microsoft-MIEngine-Out-4vmlzlhi.25j' '-
soft-MIEngine-Pid-4fawwso4.qrm' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the number of equations: 2
Enter the coefficients of the equations:
a[1][1] = 22
a[1][2] = 11
a[2][1] = 33
a[2][2] = 22
Enter the constant terms:
b[1] = 12
b[2] = 32

Solution:
x[1] = -0.727273
x[2] = 2.545455
Name:Amilliey Pakhrin
Roll no:32
```

**11. Write a program to solve system of nonlinear equations using Gauss Jordan method.**

```
#include <stdio.h>
#include <math.h>
#define MAX 10
void gaussJordan(float matrix[MAX][MAX], int n)
{
   int i, j, k;
   float temp;
   for (i = 0; i < n; i++)
   {
   temp = matrix[i][i];
      for (j = 0; j <= n; j++)
      {
         matrix[i][j] /= temp;
      }
      for (k = 0; k < n; k++)
      {
         if (k != i)
         {
            temp = matrix[k][i];
            for (j = 0; j <= n; j++)
            {
               matrix[k][j] -= temp * matrix[i][j];
            }
         }
      }
   }
}

int main()
{
```

```
    float matrix[MAX][MAX];
    int n, i, j;

    printf("Enter the number of equations: ");
    scanf("%d", &n);
    printf("Enter the augmented matrix (coefficients and constants):\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j <= n; j++)
        {
            scanf("%f", &matrix[i][j]);
        }
    }
    gaussJordan(matrix, n);
    printf("The solutions are:\n");
    for (i = 0; i < n; i++)
    {
        printf("x%d = %.3f\n", i + 1, matrix[i][n]);
    }
 printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensions\ms-vsco
er.exe' '--stdin=Microsoft-MIEngine-In-ess4dvy4.k03' '--stdout=Microsoft-MIEngine-Out-4ys0bi3n.
soft-MIEngine-Pid-1txjb2d3.2rl' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the number of equations: 2
Enter the augmented matrix (coefficients and constants):
4
3
5
11
12
32
The solutions are:
x1 = -2.400
x2 = 4.867
Name:Amilliey Pakhrin
Roll no:32
```

**12. Write programs to factorize matrix using Dolittle method**

```
#include <stdio.h>
#define MAX 10
void doolittleFactorization(float A[MAX][MAX], float L[MAX][MAX], float U[MAX][MAX], int n)
{
   for (int i = 0; i < n; i++)
   {
      for (int j = i; j < n; j++)
      {
         float sum = 0;
         for (int k = 0; k < i; k++)
            sum += L[i][k] * U[k][j];
         U[i][j] = A[i][j] - sum;
      }
      for (int j = i; j < n; j++)
      {
         if (i == j)
            L[i][i] = 1;
         else
         {
            float sum = 0;
            for (int k = 0; k < i; k++)
               sum += L[j][k] * U[k][i];
            L[j][i] = (A[j][i] - sum) / U[i][i];
         }
      }
   }
}

void printMatrix(float mat[MAX][MAX], int n)
{
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%0.2f\t", mat[i][j]);
        }
        printf("\n");
    }
}

int main()
{
    int n;
    float A[MAX][MAX], L[MAX][MAX] = {0}, U[MAX][MAX] = {0};

    printf("Enter the size of the matrix (n x n): ");
    scanf("%d", &n);

    printf("Enter the elements of the matrix A:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%f", &A[i][j]);
        }
    }

    doolittleFactorization(A, L, U, n);

    printf("\nMatrix A:\n");
    printMatrix(A, n);
```

```
    printf("\nLower Triangular Matrix L:\n");
    printMatrix(L, n);

    printf("\nUpper Triangular Matrix U:\n");
    printMatrix(U, n);
            printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensions\ms-vsco
er.exe' '--stdin=Microsoft-MIEngine-In-xcd1tqbl.wnt' '--stdout=Microsoft-MIEngine-Out-e5i2ud53.
soft-MIEngine-Pid-fxe0cp2m.hjh' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the size of the matrix (n x n): 2 x2
Enter the elements of the matrix A:

Matrix A:
0.00    0.00
0.00    0.00

Lower Triangular Matrix L:
1.00    0.00
inf     1.00

Upper Triangular Matrix U:
0.00    0.00
0.00    -nan(ind)
Name:Amilliey Pakhrin
Roll no:32
```

**13. Write programs to factorize matrix using Cholesky method**

```
#include <stdio.h>
#include <math.h>
#define MAX 10
void choleskyFactorization(float A[MAX][MAX], float L[MAX][MAX], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <= i; j++)
        {
```

```
            float sum = 0;

            if (j == i)
            {
                for (int k = 0; k < j; k++)
                    sum += L[j][k] * L[j][k];
                L[j][j] = sqrt(A[j][j] - sum);
            }
            else
            {
                for (int k = 0; k < j; k++)
                    sum += L[i][k] * L[j][k];
                L[i][j] = (A[i][j] - sum) / L[j][j];
            }
        }
    }
}

void printMatrix(float mat[MAX][MAX], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i >= j)
                printf("%0.2f\t", mat[i][j]);
            else
                printf("0.00\t");
        }
        printf("\n");
    }
```

```
}
int main()
{
   int n;
   float A[MAX][MAX], L[MAX][MAX] = {0};
   printf("Enter the size of the matrix (n x n): ");
   scanf("%d", &n);
   printf("Enter the elements of the matrix A (symmetric and positive definite):\n");
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         scanf("%f", &A[i][j]);
      }
   }

   int isSymmetric = 1;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         if (A[i][j] != A[j][i])
         {
            isSymmetric = 0;
            break;
         }
      }
      if (!isSymmetric)
         break;
   }
   if (!isSymmetric)
```

```
    {
        printf("Matrix is not symmetric. Cholesky factorization requires a symmetric positive definite matrix.\n");
        return 1;
    }
    choleskyFactorization(A, L, n);
    printf("\nMatrix A:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%0.2f\t", A[i][j]);
        }
        printf("\n");
    }
    printf("\nLower Triangular Matrix L:\n");
    printMatrix(L, n);
printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
Enter the size of the matrix (n x n): 2x2
Enter the elements of the matrix A (symmetric and positive definite):

Matrix A:
0.00    0.00
0.00    0.00

Lower Triangular Matrix L:
0.00    0.00
0.00    0.00
Name:Amilliey Pakhrin
Roll no:32
```

**14. Write programs to solve system of non-linear equations using Jacobi Iteration.**

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

```
#define MAX 10
#define EPSILON 0.0001
#define MAX_ITER 100
void jacobiIteration(double equations[MAX][MAX + 1], double x[MAX], int n)
{
    double new_x[MAX];
    int iter = 0;
    double max_diff;
    do
    {
        max_diff = 0.0;
        for (int i = 0; i < n; i++)
        {
            new_x[i] = equations[i][n];
            for (int j = 0; j < n; j++)
            {
                if (i != j)
                {
                    new_x[i] -= equations[i][j] * x[j];
                }
            }
            new_x[i] /= equations[i][i];
            double diff = fabs(new_x[i] - x[i]);
            if (diff > max_diff)
                max_diff = diff;
        }
        for (int i = 0; i < n; i++)
        {
            x[i] = new_x[i];
        }
        iter++;
```

```
    } while (max_diff > EPSILON && iter < MAX_ITER);
    if (iter >= MAX_ITER)
    {
        printf("Solution did not converge within the maximum number of iterations.\n");
    }
    else
    {
        printf("Solution converged in %d iterations.\n", iter);
    }
}
int main()
{
    int n;
    double equations[MAX][MAX + 1], x[MAX] = {0};

    printf("Enter the number of variables (n): ");
    scanf("%d", &n);

    printf("Enter the coefficients of the equations row-wise (including the constant term):\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <= n; j++)
        {
            scanf("%lf", &equations[i][j]);
        }
    }
    printf("Enter the initial guesses for the variables:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &x[i]);
    }
```

```
    jacobiIteration(equations, x, n)
    printf("The solutions are:\n");
    for (int i = 0; i < n; i++)
    {
        printf("x[%d] = %.4f\n", i + 1, x[i]);
    }
                printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
Enter the number of variables (n): 1
Enter the coefficients of the equations row-wise (including the constant term):
2
3
Enter the initial guesses for the variables:
33
Solution converged in 2 iterations.
The solutions are:
x[1] = 1.5000
Name:Amilliey Pakhrin
Roll no:32
```

**15. Write programs to solve system of non-linear equations using Gauss-Seidel method.**

```
#include <stdio.h>
#include <math.h>
#define MAX 10
#define EPSILON 0.0001
#define MAX_ITER 100
void gaussSeidelIteration(double equations[MAX][MAX + 1], double x[MAX], int n)
{
    int iter = 0;
    double max_diff;
    do
    {
        max_diff = 0.0;
        for (int i = 0; i < n; i++)
```

```
        {
            double sum = equations[i][n];
            for (int j = 0; j < n; j++)
            {
                if (i != j)
                {
                    sum -= equations[i][j] * x[j];
                }
            }
            double new_x = sum / equations[i][i];
            double diff = fabs(new_x - x[i]);
            if (diff > max_diff)
            {
                max_diff = diff;
            }
            x[i] = new_x;
        }
        iter++;
    } while (max_diff > EPSILON && iter < MAX_ITER);
    if (iter >= MAX_ITER)
    {
        printf("Solution did not converge within the maximum number of iterations.\n");
    }
    else
    {
        printf("Solution converged in %d iterations.\n", iter);
    }
}
int main()
{
    int n;
```

```
    double equations[MAX][MAX + 1], x[MAX] = {0};
    printf("Enter the number of variables (n): ");
    scanf("%d", &n);
    printf("Enter the coefficients of the equations row-wise (including the constant term):\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <= n; j++)
        {
            scanf("%lf", &equations[i][j]);
        }
    }
    printf("Enter the initial guesses for the variables:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &x[i]);
    }
    gaussSeidelIteration(equations, x, n);
    printf("The solutions are:\n");
    for (int i = 0; i < n; i++)
    {
        printf("x[%d] = %.4f\n", i + 1, x[i]);
    }
                printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
Enter the number of variables (n): 2
Enter the coefficients of the equations row-wise (including the constant term):
2
4
23
43
11
23
Enter the initial guesses for the variables:
45
32
Solution did not converge within the maximum number of iterations.
The solutions are:
x[1] = -1344804698946668164105231064437100626205994010266578474745215398725975240739556641197785088.0000
x[2] = 525696382315515710149921999814216386365915465502240582519191434038063536704074696550776832O.0000
name:Amilliey Pakhrin
Roll no:32
```

**16. Write a program to find eigenvalue and eigenvector using power method.**

```
#include <stdio.h>
#include <math.h>
#define MAX 10
#define EPSILON 0.0001
#define MAX_ITER 100
void powerMethod(double matrix[MAX][MAX], double vector[MAX], int n)
{
   double new_vector[MAX], lambda_old = 0.0, lambda_new = 0.0;
   int iter = 0;
   double norm = 0.0;
   for (int i = 0; i < n; i++)
   {
      norm += vector[i] * vector[i];
   }
   norm = sqrt(norm);
   for (int i = 0; i < n; i++)
   {
      vector[i] /= norm;
   }
   do
   {
      for (int i = 0; i < n; i++)
      {
         new_vector[i] = 0.0;
         for (int j = 0; j < n; j++)
         {
            new_vector[i] += matrix[i][j] * vector[j];
         }
      }
      lambda_new = 0.0;
```

```
    for (int i = 0; i < n; i++)
    {
        lambda_new += new_vector[i] * vector[i];
    }
    norm = 0.0;
    for (int i = 0; i < n; i++)
    {
        norm += new_vector[i] * new_vector[i];
    }
    norm = sqrt(norm);
    for (int i = 0; i < n; i++)
    {
        new_vector[i] /= norm;
    }
    double diff = fabs(lambda_new - lambda_old);
    if (diff < EPSILON)
    {
        break;
    }
    for (int i = 0; i < n; i++)
    {
        vector[i] = new_vector[i];
    }
    lambda_old = lambda_new;
    iter++;
} while (iter < MAX_ITER);

if (iter >= MAX_ITER)
{
    printf("The Power Method did not converge within the maximum number of iterations.\n");
}
```

```
    else
    {
        printf("The Power Method converged in %d iterations.\n", iter);
        printf("Dominant Eigenvalue: %.6f\n", lambda_new);
        printf("Corresponding Eigenvector:\n");
        for (int i = 0; i < n; i++)
        {
            printf("x[%d] = %.6f\n", i + 1, vector[i]);
        }
    }
}

int main()
{
    int n;
    double matrix[MAX][MAX], vector[MAX];

    printf("Enter the size of the square matrix (n): ");
    scanf("%d", &n);

    printf("Enter the elements of the matrix row-wise:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%lf", &matrix[i][j]);
        }
    }
    printf("Enter the initial guess vector:\n");
    for (int i = 0; i < n; i++)
    {
```

```
    scanf("%lf", &vector[i]);
  }
  powerMethod(matrix, vector, n);
        printf("Name:Amilliey Pakhrin \n roll no.32");
  return 0;
}
```

```
Enter the size of the square matrix (n): 1
Enter the elements of the matrix row-wise:
12
Enter the initial guess vector:
2
The Power Method converged in 1 iterations.
Dominant Eigenvalue: 12.000000
Corresponding Eigenvector:
x[1] = 1.000000
Name:Amilliey Pakhrin
Roll no:32
```

**Lab-5**

**17. Write programs to implement Euler's method method to solve ordinary differential equations.**

```
#include <stdio.h>
double f(double x, double y)
{
  return x + y;
}
void eulerMethod(double x0, double y0, double xn, double h)
{
  double x = x0;
  double y = y0;
  printf("x\t\t y\n");
  printf("%.4f\t %.4f\n", x, y);
  while (x < xn)
  {
    y = y + h * f(x, y);
    x = x + h;
    printf("%.4f\t %.4f\n", x, y);
```

```
    }
}
int main()
{
    double x0, y0, xn, h;
    printf("Enter the initial value of x (x0): ");
    scanf("%lf", &x0);

    printf("Enter the initial value of y (y0): ");
    scanf("%lf", &y0);

    printf("Enter the final value of x (xn): ");
    scanf("%lf", &xn);

    printf("Enter the step size (h): ");
    scanf("%lf", &h);
    printf("\nSolving the ODE using Euler's method:\n");
    eulerMethod(x0, y0, xn, h);
                printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
Enter the initial value of x (x0): 32
Enter the initial value of y (y0): 22
Enter the final value of x (xn): 12
Enter the step size (h): 2

Solving the ODE using Euler's method:
x               y
32.0000  22.0000
Name:Amilliey Pakhrin
Roll no:32
```

**18. Write programs to implement Heun's method to solve ordinary differential equations.**

```
#include <stdio.h>
double f(double x, double y)
```

```
{
    return x + y;
}
void heunsMethod(double x0, double y0, double xn, double h)
{
    double x = x0, y = y0, y_predict, slope;
    printf("x\t\t y\n");
    printf("%.4f\t %.4f\n", x, y);
    while (x < xn)
    {
        y_predict = y + h * f(x, y);
        slope = (f(x, y) + f(x + h, y_predict)) / 2.0;
        y = y + h * slope;
        x = x + h;
        printf("%.4f\t %.4f\n", x, y);
    }
}
int main()
{
    double x0, y0, xn, h;
    printf("Enter the initial value of x (x0): ");
    scanf("%lf", &x0);
    printf("Enter the initial value of y (y0): ");
    scanf("%lf", &y0);
    printf("Enter the final value of x (xn): ");
    scanf("%lf", &xn);

    printf("Enter the step size (h): ");
    scanf("%lf", &h);

    printf("\nSolving the ODE using Heun's method:\n");
```

```
    heunsMethod(x0, y0, xn, h);
                printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
Enter the initial value of x (x0): 12
Enter the initial value of y (y0): 23
Enter the final value of x (xn): 11
Enter the step size (h): 3

Solving the ODE using Heun's method:
x               y
12.0000  23.0000
Name:Amilliey Pakhrin
Roll no:32
```

**19. Write a program to solve boundary value problem using shooting method.**

```
#include <stdio.h>
#include <math.h>
double f(double x, double y, double z)
{
    return -2 * y;
}
double g(double x, double y, double z)
{
    return z;
}
void euler(double x0, double y0, double z0, double h, double xn, double *y_end)
{
    double x = x0, y = y0, z = z0;
    while (x < xn)
    {
        double k1 = h * g(x, y, z);
        double l1 = h * f(x, y, z);
        y += k1;
        z += l1;
        x += h;
```

```
    }
    *y_end = y;
}
void shootingMethod(double x0, double y0, double xn, double yn, double z_guess1, double z_guess2, double h)
{
    double y_end1, y_end2;
    double z1 = z_guess1, z2 = z_guess2;

    euler(x0, y0, z1, h, xn, &y_end1);

    euler(x0, y0, z2, h, xn, &y_end2);

    double z_new, y_end_new;
    while (fabs(y_end1 - yn) > 1e-6)
    {
        z_new = z1 + (yn - y_end1) * (z2 - z1) / (y_end2 - y_end1);
        euler(x0, y0, z_new, h, xn, &y_end_new);
        z1 = z2;
        y_end1 = y_end2;
        z2 = z_new;
        y_end2 = y_end_new;
    }
    printf("Final solution: z(x0) = %.6f, y(xn) = %.6f\n", z_new, y_end_new);
}
int main()
{
    double x0 = 0, y0 = 1;
    double xn = 1, yn = 0;
    double z_guess1 = -1, z_guess2 = 0;
    double h = 0.1;
```

```
    printf("Solving boundary value problem using Shooting Method...\n");
    shootingMethod(x0, y0, xn, yn, z_guess1, z_guess2, h);
                printf("Name:Amilliey Pakhrin \n roll no.32");

    return 0;
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensi
er.exe' '--stdin=Microsoft-MIEngine-In-cuyre2i0.0rj' '--stdout=Microsoft-MIEngine-Ou
soft-MIEngine-Pid-e3e5zqnv.05r' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpret
Solving boundary value problem using Shooting Method...
Final solution: z(x0) = -0.035942, y(xn) = -0.000000
Name:Amilliey Pakhrin
Roll no:32
```

**Lab-6**

**20. Write programs to solve Laplacian Equation.**

```
#include <stdio.h>
#include <math.h>
#define MAX 100
#define TOLERANCE 1e-6
void solveLaplace(double u[MAX][MAX], int rows, int cols, int maxIterations)
{
    int i, j, iter;
    double diff, maxDiff;
    for (iter = 0; iter < maxIterations; iter++)
    {
        maxDiff = 0.0;
        for (i = 1; i < rows - 1; i++)
        {
            for (j = 1; j < cols - 1; j++)
            {
                double oldVal = u[i][j];
                u[i][j] = 0.25 * (u[i + 1][j] + u[i - 1][j] + u[i][j + 1] + u[i][j - 1]);
                diff = fabs(u[i][j] - oldVal);
```

```
            if (diff > maxDiff)
            {
                maxDiff = diff;
            }
        }
    }
    if (maxDiff < TOLERANCE)
    {
        printf("Converged after %d iterations.\n", iter + 1);
        break;
    }
  }
  if (iter == maxIterations)
  {
    printf("Reached maximum iterations without full convergence.\n");
  }
}
int main()
{
  int rows = 5, cols = 5;
  double u[MAX][MAX] = {0};
  int i, j;
  for (i = 0; i < rows; i++)
  {
    u[i][0] = 100.0;
    u[i][cols - 1] = 100.0;
  }
  for (j = 0; j < cols; j++)
  {
    u[0][j] = 0.0;
    u[rows - 1][j] = 0.0;
```

```
    }
    printf("Initial Grid:\n");
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            printf("%6.2f ", u[i][j]);
        }
        printf("\n");
    }
    solveLaplace(u, rows, cols, 10000);
    printf("\nSolution Grid:\n");
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            printf("%6.2f ", u[i][j]);
        }
        printf("\n");
    }
                printf("Name:Amilliey Pakhrin \n roll no.32");
    return 0;
}
```

```
PS C:\Users\Lenovo\Desktop\Numerical-Method-Lab>  & 'c:\Users\Lenovo\.vscode\extensions\ms-vsc
er.exe' '--stdin=Microsoft-MIEngine-In-cqydjvhh.i4z' '--stdout=Microsoft-MIEngine-Out-5ov0et53
soft-MIEngine-Pid-jnd5epzy.aie' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Initial Grid:
  0.00   0.00   0.00   0.00   0.00
100.00   0.00   0.00   0.00 100.00
100.00   0.00   0.00   0.00 100.00
100.00   0.00   0.00   0.00 100.00
  0.00   0.00   0.00   0.00   0.00
Converged after 27 iterations.

Solution Grid:
  0.00   0.00   0.00   0.00   0.00
100.00  50.00  37.50  50.00 100.00
100.00  62.50  50.00  62.50 100.00
100.00  50.00  37.50  50.00 100.00
  0.00   0.00   0.00   0.00   0.00
Name:Amilliey Pakhrin
Roll no:32
```

**21. Write programs to solve Poisson's Equation.**

```
#include <stdio.h>
#include <math.h>
#define MAX 100
#define TOLERANCE 1e-6
double sourceTerm(double x, double y)
void solvePoisson(double u[MAX][MAX], double f[MAX][MAX], int rows, int cols, double h, int maxIterations)
{
   int i, j, iter;
   double diff, maxDiff;
   for (iter = 0; iter < maxIterations; iter++)
   {
      maxDiff = 0.0;
      for (i = 1; i < rows - 1; i++)
      {
         for (j = 1; j < cols - 1; j++)
         {
            double oldVal = u[i][j];
            u[i][j] = 0.25 * (u[i + 1][j] + u[i - 1][j] + u[i][j + 1] + u[i][j - 1] - h * h * f[i][j]);
            diff = fabs(u[i][j] - oldVal);
            if (diff > maxDiff)
            {
               maxDiff = diff;
            }
         }
      }
      if (maxDiff < TOLERANCE)
      {
         printf("Converged after %d iterations.\n", iter + 1);
         break;
      }
```

```
    }
    if (iter == maxIterations)
    {
        printf("Reached maximum iterations without full convergence.\n");
    }
}
int main()
{
    int rows = 5, cols = 5;
    double u[MAX][MAX] = {0};
    double f[MAX][MAX] = {0};
    int i, j;
    double x, y;
    double h = 1.0;
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            x = i * h;
            y = j * h;
            f[i][j] = sourceTerm(x, y);
        }
    }
    for (i = 0; i < rows; i++)
    {
        u[i][0] = 0.0;
        u[i][cols - 1] = 0.0;
    }
    for (j = 0; j < cols; j++)
    {
        u[0][j] = 0.0;
```

```
    u[rows - 1][j] = 0.0;
  }
  printf("Initial Grid:\n");
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      printf("%6.2f ", u[i][j]);
    }
    printf("\n");
  }
  solvePoisson(u, f, rows, cols, h, 10000);
  printf("\nSolution Grid:\n");
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      printf("%6.2f ", u[i][j]);
    }
    printf("\n");
  }
            printf("Name:Amilliey Pakhrin \n roll no.32");
  return 0;
}
```

```
Initial Grid:
  0.00   0.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00
Converged after 25 iterations.

Solution Grid:
  0.00   0.00   0.00   0.00   0.00
  0.00  -3.64  -6.29  -6.50   0.00
  0.00  -6.29 -10.00  -9.71   0.00
  0.00  -6.50  -9.71  -9.36   0.00
  0.00   0.00   0.00   0.00   0.00
Name:Amilliey Pakhrin
Roll no:32
```