Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: Newton Forward Interpolation

CODE:

```
#include <bits/stdc++.h>
using namespace std;

int fact(int n) {
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}

int main() {
    cout << "input total number of values:";
    int n;
    cin >> n;
    vector<vector<float>> mat(n, vector<float>(n + 1, 0));
    cout << endl
         << "input values of x:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> mat[i][0];
    }

    cout << endl
         << "input values of y:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> mat[i][1];
    }
    float y;
    cout << endl
         << "input value to find:";
    cin >> y;
    cout << endl;
    for (int i = 2; i < n + 1; i++) {
        for (int j = 0; j < n - i + 1; j++)
            mat[j][i] = mat[j + 1][i - 1] - mat[j][i - 1];
    }

    cout << "Interpolation table" << endl;
    cout << setw(4) << "X"
         << "\t"
         << "   Y"
         << "\t"
         << endl;
    for (int i = 0; i < n; i++) {
        cout << setw(4) << mat[i][0]
             << "\t";
        cout << setw(4) << mat[i][1]
```

```
            << "\t";
        for (int j = 2; j < n - i + 1; j++)
            cout << setw(4) << mat[i][j]
                 << "\t";
        cout << endl;
    }

    float sum = mat[0][1];
    float u = (y - mat[0][0]) / (mat[1][0] - mat[0][0]);
    for (int i = 1; i < n; i++) {
        // u series calculation for each i
        float temp = u;
        for (int j = 1; j < i; j++)
            temp = temp * (u - j);
        sum = sum + (temp * mat[0][i + 1]) / fact(i);
    }

    cout << "\n Value at " << y << " is "
         << sum << endl;
    return 0;
}
```

OUTPUT:

```
input total number of values:5

input values of x:
1891
1901
1911
1921
1931

input values of y:
46
66
81
93
101

input value to find:1925

   X       Y
1891      46      20      -5       2      -3
1901      66      15      -3      -1
1911      81      12      -4
1921      93       8
1931     101

 Value at 1925 is 96.8368
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: Newton Backward Interpolation

CODE:

```
#include <bits/stdc++.h>
using namespace std;

int fact(int n) {
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}

int main() {
    cout << "input total number of values:";
    int n;
    cin >> n;
    vector<vector<float>> mat(n, vector<float>(n + 1, 0));
    cout << endl
         << "input values of x:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> mat[i][0];
    }

    cout << endl
         << "input values of y:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> mat[i][1];
    }
    float y;
    cout << endl
         << "input value to find:";
    cin >> y;
    cout << endl;
    for (int i = 2; i < n + 1; i++) {
        for (int j = 0; j < n - i + 1; j++)
            mat[j][i] = mat[j + 1][i - 1] - mat[j][i - 1];
    }

    cout << "Interpolation table" << endl;
    cout << setw(4) << "X"
         << "\t"
         << "   Y"
         << "\t"
         << endl;
    for (int i = 0; i < n; i++) {
        cout << setw(4) << mat[i][0]
             << "\t";
        cout << setw(4) << mat[i][1]
```

```
            << "\t";
        for (int j = 2; j < n - i + 1; j++)
            cout << setw(4) << mat[i][j]
                 << "\t";
        cout << endl;

    }

    float sum = mat[n - 1][1];
    float u = (y - mat[n - 1][0]) / (mat[1][0] - mat[0][0]);
    for (int i = 1; i < n; i++) {
        // u series calculation for each i
        float temp = u;
        for (int j = 1; j < i; j++)
            temp = temp * (u + j);
        sum = sum + (temp * mat[n - 1][i + 1]) / fact(i);
    }

    cout << "\n Value at " << y << " is "
         << sum << endl;
    return 0;
}
```

OUTPUT:

```
input total number of values:5

input values of x:
1 2 3 4 5

input values of y:
5 7 9 10 14

input value to find:1.5

Interpolation table
   X       Y
   1       5       2       0      -1       5
   2       7       2      -1       4
   3       9       1       3
   4      10       4
   5      14

 Value at 1.5 is 14
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: LAGRANGE METHOD

CODE:

```
#include<stdio.h>

double lagrangeInterpolation( double xValues[], double  yValues[], double x, int n) {
   double result = 0.0;

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

      result += term;
   }

   return result;
}

int main() {

   double xValues[4] = {5.0, 6.0, 9.0,11.0};
   double yValues[4] = {12.0, 13.0, 19.0,16.0};


   double x = 10.0;
   double ans = lagrangeInterpolation(xValues, yValues, x,4);

   printf("\nans: %f",ans);



   return 0;
}
```

OUTPUT:

```
g++ langrange.cpp -o langrange } ; if ($?) { .\langrange }
ans: 18.833333
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: TRAPEZOIDAL METHOD

CODE:

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

/* Define function here */
#define f(x) 1/(1+pow(x,2))

int main()
{
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;

 /* Input */
 printf("Enter lower limit of integration: ");
 scanf("%f", &lower);
 printf("Enter upper limit of integration: ");
 scanf("%f", &upper);
 printf("Enter number of sub intervals: ");
 scanf("%d", &subInterval);

 /* Calculation */
 /* Finding step size */
 stepSize = (upper - lower)/subInterval;

 /* Finding Integration Value */
 integration = f(lower) + f(upper);
 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*stepSize;
  integration = integration + 2 * f(k);
 }
 integration = integration * stepSize/2;
 printf("\nRequired value of integration is: %.3f", integration);
 return 0;
}
```

Name: Artham Bhardwaj

University Roll: 2018718

Section: A
Class Roll.no: 21
Method Name: TRAPEZOIDAL METHOD

OUTPUT:

```
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 5

Required value of integration is: 0.784
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: SIMPSON's 1/3 METHOD

CODE:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) 1/(1+x*x)

int main()
{
 float l, u,sum=0.0,h, k;
 int i,n;

 printf("Enter lower limit of integration: ");
 scanf("%f", &l);
 printf("Enter upper limit of integration: ");
 scanf("%f", &u);
 printf("Enter number of sub intervals: ");
 scanf("%d", &n);
 h= (u-l)/h;
 sum= f(l)+f(u);
 for(i=1;i<=n-1; i++)
 {
  k =l+i*n;
  if(i%2==0)
  {
   sum=sum+2*f(k);
  }
  else
  {
   sum=sum+4*f(k);
  }
 }
 sum=sum*n/3;
 printf("\nRequired value of integration is: %.3f",sum);

 return 0;
}
```

OUTPUT:

```
BNST LAB CODES\simpsons(1\" ; if ($?) { gcc simpspn1by3.c -o simpspn1by3 } ; if (
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 5

Required value of integration is: 2.827
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: SIMPSON's 3/8 METHOD

CODE:

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

#define f(x) 1/(1+x*x)

int main()
{
 float l, u,sum=0.0, h, k;
 int i, n;
 printf("Enter lower limit of integration: ");
 scanf("%f", &l);
 printf("Enter upper limit of integration: ");
 scanf("%f", &u);
 printf("Enter number of sub intervals: ");
 scanf("%d", &n);
h= (u - l)/n;
 sum = f(l)+f(u);
 for(i=1; i<= n-1; i++)
 {
  k = l+ i*h;
  if(i%3 == 0)
  {
    sum= sum+ 2 * f(k);
  }
  else
  {
   sum=sum+3*f(k);
  }
 }
 sum=sum*h*3/8;
 printf("\nRequired value of integration is: %.3f",sum);

 return 0;
}
```

OUTPUT:

```
CBNST LAB\CBNST LAB CODES\simpsons(1\" ; if ($?) { gcc simpson3by8.c -o si
Enter lower limit of integration: 0
Enter upper limit of integration: 2
Enter number of sub intervals: 6

Required value of integration is: 1.106
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70

Method Name: RANGA KUTTA

METHOD

CODE:

```
#include <iostream>
using namespace std;

// A sample differential equation "dy/dx = 1+(y/x), 1<=x<=3"
float dydx(float x, float y) {
    return (1 + (y / x));
}

// Finds value of y for a given x using step size h
// and initial value y0 at x0.
float rungeKutta(float x0, float y0, float x, float h) {
    // Count number of iterations using step size or
    // step height h
    int n = static_cast<int>((x - x0) / h);

    float k1, k2, k3, k4;

    // Iterate for number of iterations
    float y = y0;
    for (int i = 1; i <= n; i++) {
        // Apply Runge Kutta Formulas to find
        // the next value of y
        k1 = h * dydx(x0, y);
        k2 = h * dydx(x0 + 0.5 * h, y + 0.5 * k1);
        k3 = h * dydx(x0 + 0.5 * h, y + 0.5 * k2);
        k4 = h * dydx(x0 + h, y + k3);

        // Update the next value of y
        y = y + (1.0 / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4);

        // Update the next value of x
        x0 = x0 + h;
        cout<<"y: "<<y<<"  x: "<<x0<<endl;
    }

    return y;
}

// Driver Code
int main() {
```

```
    float x0 = 1, y = 1, x = 3, h = 1;
    cout << "The value of y at x is: " << rungeKutta(x0, y, x, h) << endl;

    return 0;
}
```

OUTPUT:

```
" ; if ($?) { g++ ranga_kutta.cpp -o ranga_kutta
y: 3.37963  x: 2
y: 6.285  x: 3
The value of y at x is: 6.285
```

Name: VEDANT KASHYAP
University Roll: 2019224
Section: A
Class Roll.no: 70
Method Name: EULER' S METHOD

CODE:

```
#include <iostream>
using namespace std;

// Consider a differential equation
// dy/dx=1+(y/x); 1<=x<=3
float func(float x, float y)
{
   return 1+(y/x);
}

// Function for Euler formula
void euler(float x0, float y, float h, float x)
{
   float temp = 0;

   // Iterating till the point at which we
   // need approximation
   while (x0 < x) {
      temp = y;
      y = y + h * func(x0, y);
      x0 = x0 + h;
      cout<<"y: "<<y<<" x: "<<x0<<endl;
   }


}

// Driver program
int main()
{
   // Initial Values
   float x0 = 1;
   float y0 = 1;
   float h = 1;

   // Value of x at which we need approximation
   float x = 3;

   euler(x0, y0, h, x);
   return 0;
}
```

OUTPUT: