Aditya Bhawalkar

FINIQ ASSIGNMENT PROGRAMS

WEEK 1

The main objective of the assignment is broadly divided into the following three parts:

1. Numerical methods of integration
2. Numerical methods of differentiation
3. Random number function

1. Numerical Methods of Integration:

This part is broadly called as the Riemann Sum method that mainly deals with the approximation of the integral by a finite sum.

It can be done in the following 5 ways:

1. Left Riemann Sum (Done)
2. Right Riemann Sum (Done)
3. Trapezoidal Rule (Done)
4. Mid-Point Rule (Done)
5. Simpson's Rule (Done)

2. Numerical Methods of Differentiation:

Following are the major methods of differentiation:

1. Central Difference
2. Forward Difference
3. Backward Difference
4. Newton's Interpolation (Done)
5. Lagrange's Interpolation (Done)

3. Random Number Generator:

The method must generate a random number and return it. It takes the min and ax value as a parameter.

Reimann Sum(Left, Right, Mid-Point):

```
import numpy as np
import matplotlib.pyplot as plt
import math
def right_riemann(f, a, b, N):
    f = np.vectorize(f)
    n = N # Use n*N+1 points to plot the function smoothly


    x = np.linspace(a,b,N+1)
    y = f(x)
```

```
    X = np.linspace(a,b,n*N+1)
    Y = f(X)

    plt.figure(figsize=(15,5))
    plt.plot(X,Y,'b')
    x_right = x[1:] # Left endpoints
    y_right = y[1:]
    plt.plot(x_right,y_right,'b.',markersize=10)
    plt.bar(x_right,y_right,width=-(b-a)/N,alpha=0.2,align='edge',edgecolor='b')
    plt.title('Right Riemann Sum, N = {}'.format(N))
    plt.show()
    dx = (b-a)/N
    x_right = np.linspace(dx,b,N)
    right_riemann_sum = np.sum(f(x_right) * dx)
    return right_riemann_sum

def left_riemann(f, a, b, N):
    f = np.vectorize(f)
    n = N # Use n*N+1 points to plot the function smoothly


    x = np.linspace(a,b,N+1)
    y = f(x)

    X = np.linspace(a,b,n*N+1)
    Y = f(X)

    plt.figure(figsize=(15,5))
    plt.plot(X,Y,'b')
    x_left = x[:-1] # Left endpoints
    y_left = y[:-1]
    plt.plot(x_left,y_left,'b.',markersize=10)
    plt.bar(x_left,y_left,width=(b-a)/N,alpha=0.2,align='edge',edgecolor='b')
    plt.title('Left Riemann Sum, N = {}'.format(N))

    plt.show()
    dx = (b-a)/N
    x_left = np.linspace(a,b-dx,N)
    left_riemann_sum = np.sum(f(x_left) * dx)
    return left_riemann_sum



def midpoint_riemann(f, a, b, N):
    f = np.vectorize(f)
```

```
    n = N # Use n*N+1 points to plot the function smoothly


    x = np.linspace(a,b,N+1)
    y = f(x)

    X = np.linspace(a,b,n*N+1)
    Y = f(X)

    plt.figure(figsize=(15,5))
    plt.plot(X,Y,'b')
    x_mid = (x[:-1] + x[1:])/2 # Midpoints
    y_mid = f(x_mid)
    plt.plot(x_mid,y_mid,'b.',markersize=10)
    plt.bar(x_mid,y_mid,width=(b-a)/N,alpha=0.2,edgecolor='b')
    plt.title('Midpoint Riemann Sum, N = {}'.format(N))
    plt.show()
    dx = (b-a)/N
    x_midpoint = np.linspace(dx/2,b - dx/2,N)
    midpoint_riemann_sum = np.sum(f(x_midpoint) * dx)
    return(midpoint_riemann_sum)


ans = midpoint_riemann(lambda x : 6*x**2, 0, 5, 10)
print(ans)
```

Trapezoidal Integration:

```
import numpy as np
import matplotlib.pyplot as plt

def trapezoidal(f, a, b, N):
    f = np.vectorize(f)
    n = N
    x = np.linspace(a, b, N+1)
    y=f(x)

    X = np.linspace(a, b, n*N+1)
    Y= f(X)
```

```
    plt.plot(X, Y)
    for i in range(N):
        xs = [x[i], x[i], x[i+1], x[i+1]]
        ys = [0, f(x[i]), f(x[i+1]), 0]
        plt.fill(xs, ys, 'b', edgecolor = 'b', alpha = 0.2)

    plt.title('Trapezoidal Rule, N = {}'.format(N))
    plt.show()

    y = f(x)
    y_right = y[1:] # Right endpoints
    y_left = y[:-1] # Left endpoints
    dx = (b - a)/N
    T = (dx/2) * np.sum(y_right + y_left)
    return T

ans = trapezoidal(lambda x : 6*x**2, 2, 4, 10)
print(ans)
```

Simpson's Integration:

```
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt

def simpsons(f, a, b, N = 50):
    #Deciding and validating the step size
    if N % 2 == 1:
        raise ValueError("N must be an even number")
    dx = (b - a)/ N
    x = np.linspace(a, b, N+1)
    y = f(x)
    S = dx / 3 * np.sum(y[0:-1:2] + 4 * y[1::2] + y[2::2])

    integrals = []
    x_range = []
    y_range = []
    for i in x:
```

```
        x_range.append(i)

        y_range.append(f(i))
        integral = integrate.simps(y_range, x_range)
        integrals.append(integral)
    #plotting the output
    plt.plot(x, integrals)
    plt.show()

    return(S)


#pass the function as a lambda function
print(simpsons(lambda x :6*x**2 , 2, 4, 100))
```

Generic Integration Program for Testing:

```
from fractions import Fraction

def left_rect(f, x, h):
    return f(x)

def mid_rect(f, x, h):
    return f(x + h / 2)

def right_rect(f, x, h):
    return f(x+h)

def trapezium(f, x, h):
    return (f(x) + f(x + h))/2.0

def simpson(f, x, h):
    return (f(x) + 4*f(x + h/2) + f(x + h))/6.0

def cube(x):
    return x*x*x

def reciprocal(x):
    return 1/x
```

```
def identity(x):
    return x

def raiseToFour(x):
    return x*x*x*x

def integrate(f, a, b, steps, meth):
    h = (b - a) / steps
    ival = h * sum(meth(f, a+i*h, h) for i in range(steps))
    return ival

# Tests
for a, b, steps, func in ((0., 1., 100, cube), (1., 100., 1000, reciprocal)):
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               integrate( func, a, b, steps, rule)))
    a, b = Fraction.from_float(a), Fraction.from_float(b)
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps and fractions) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               float(integrate( func, a, b, steps, rule))))

# Extra tests (compute intensive)
for a, b, steps, func in ((0., 5000., 5000000, identity),
                          (0., 6000., 6000000, identity)):
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               integrate( func, a, b, steps, rule)))
    a, b = Fraction.from_float(a), Fraction.from_float(b)
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps and fractions) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               float(integrate( func, a, b, steps, rule))))


#Testing the raiseToFour function

for a, b, steps, func in ((0., 1., 100, raiseToFour), (1., 100., 1000, reciprocal)):
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               integrate( func, a, b, steps, rule)))
```

```
    a, b = Fraction.from_float(a), Fraction.from_float(b)
    for rule in (left_rect, mid_rect, right_rect, trapezium, simpson):
        print('%s integrated using %s\n  from %r to %r (%i steps and fractions) = %r' %
              (func.__name__, rule.__name__, a, b, steps,
               float(integrate( func, a, b, steps, rule))))
```

Forward, Backward, Central Differentiation:

```
import numpy as np
import matplotlib.pyplot as plt
def derivative(f,a,method='central',h=0.01):

    if method == 'central':
        #central method
        return (f(a + h) - f(a - h))/(2*h)
    elif method == 'forward':
        #forward method
        return (f(a + h) - f(a))/h
    elif method == 'backward':
        #backward method
        return (f(a) - f(a - h))/h
    else:
        raise ValueError("Method must be 'central', 'forward' or 'backward'.")

x = np.linspace(0,5*np.pi,100)
dydx = derivative(np.sin,x)

dYdx = np.cos(x)
#plot the output using matplotlib
plt.figure(figsize=(12,5))
plt.plot(x,dydx,'r.',label='Central Difference')
plt.plot(x,dYdx,'b',label='True Value')

plt.title('Central Difference Derivative of y = cos(x)')
plt.legend(loc='best')
plt.show()


x = np.linspace(0,6,100)
```

```
f = lambda x: ((4*x**2 + 2*x + 1)/(x + 2*np.exp(x)))**x
y = f(x)
dydx = derivative(f,x)

plt.figure(figsize=(12,5))
plt.plot(x,y,label='y=f(x)')
plt.plot(x,dydx,label="Central Difference y=f'(x)")
plt.legend()
plt.grid(True)

plt.show()
```

Richardson's Differentiation:

```
from math import *
def zeros(n,m): # Zeros matrix for preallocation
    Z=[]
    for i in range(n):
        Z.append([0]*m)
    return Z

def D(Func,a,h):    # centered finite difference with step size h at point x=a
    return (Func(a+h)-Func(a-h))/(2*h)

def Richardson_dif(func,a):
    '''Richardson extrapolation method for numerical calculation of first derivative '''
    k=9 # you can change the order of approximation but try keeping it under 10 to
circumvent round-off errors.
    L=zeros(k,k)
    for I in range(k):
        L[I][0]=D(func,a,1/(2**(I+1)))
    for j in range(1,k):
        for i in range(k-j):
            L[i][j]=((4**(j))*L[i+1][j-1]-L[i][j-1])/(4**(j)-1)
    return L[0][k-1]
print('Numerical differentiation of Func=-0.1*x**4-0.15*x**3-0.5*x**2-0.25*x+1.2 at
x=0.5')
print('%04.20f'%Richardson_dif(lambda x: -0.1*x**4-0.15*x**3-0.5*x**2-0.25*x+1.2 ,
0.5))
print('diff(2**cos(pi+sin(x)) at x=pi/2 is equal to = %04.20f'%Richardson_dif(lambda x:
2**cos(pi+sin(x)),pi/3))
```

Random Number Generator (Blum Blum Shub):

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


int * rand_bbs(int l);
int main()
{
   int *p;
   //int r = 0;
   p = rand_bbs(10);
   //for(r = 0; r < 10; r++)
   //    printf("%d ",p[r]);
   return 0;
}

int * rand_bbs(int l) {
   unsigned long int seed = time(NULL) * 1000000000;
   int p = 7727;
   int q = 5527;
   int n = p * q;
   static int r[5];
   unsigned long int bold, bnew;
   if(seed == n)
      seed = 12312;
   int i, j, k = 0;
   bold = seed;
   //printf("RANDOM NUMBERS\n");
   for(i = 0; i < l; i++) {
      unsigned int bit32 = 0, bit4;
      for(j = 0; j < 4; j++) {
         bnew = (bold * bold) % n;
         //printf("%d\n", bnew);
         bit4 = bnew % 32;
         bit32 = bit32 << 4;
         bit32 = bit32 | bit4;
         bold = bnew;
      }
```

```
        //printf("%d\n", bit32);
        r[k++] = bit32;
    }
    return r;
}
```

Random Number Generator (Linear Fibonacci Generator):

```
#include<stdio.h>
int * lfg() {
    int j = 3;
    int k = 7;
    int s[] = {8, 6, 7, 5, 3, 0, 9};
    int z = sizeof(s) / sizeof(s[0]);
    int out;
    int i;
    int n;
    int a = 0;
    static int t[10];
    for(n = 0; n < 10; n++) {
        for (i = 0; i < z; i++) {
            if (i == 0) {
                out = (s[j - 1] + s[k - 1]) % 10; //the pseudorandom output
            }

            else if (i > 0 && i < 6) {
                s[i] = s[i + 1]; //shift the array
            }

            else {
                s[i] = out;
                t[a++] = s[i];
            }
        }
    }
    return t;
}
int main()
{
    int *p;
    p = lfg();
    return 0;
}
```

Random Number Generator (Linear Congruential Generator):

```
using namespace std;
class mRND
{
public:
   void seed( unsigned int s ) { _seed = s; }

protected:
   mRND() : _seed( 0 ), _a( 0 ), _c( 0 ), _m( 2147483648 ) {}
   int rnd() { return( _seed = ( _a * _seed + _c ) % _m ); }

   int _a, _c;
   unsigned int _m, _seed;
};

class MS_RND : public mRND
{
public:
   MS_RND()  { _a = 214013; _c = 2531011; }
   int rnd() { return mRND::rnd() >> 16; }
};

class BSD_RND : public mRND
{
public:
   BSD_RND() { _a = 1103515245; _c = 12345; }
   int rnd() { return mRND::rnd(); }
};

int main( int argc, char* argv[] )
{   int ll= 1 , ul = 20;
   BSD_RND bsd_rnd;
   MS_RND ms_rnd;
   int ms, bsd;
   cout << "MS RAND:" << endl << "========" << endl;
   ms = ms_rnd.rnd();
   while(ms<ll){
     ms = ll+ ms;
   }
   while (ms>=ul){
      ms = ms%ul;
   }
   while(ms<ll){
     ms = ll+ ms;
```

```
  }

   cout << ms << endl;

   cout << endl  << "BSD RAND:" << endl << "==========" << endl;
   bsd = bsd_rnd.rnd();
  while(bsd<ll){
     bsd = ll+ bsd;
   }
   while (bsd>=ul){
      bsd = bsd%ul;
   }
   while(bsd<ll){
     bsd = ll+ bsd;
  }
   cout << bsd << endl;

   cout << endl << endl;
   system( "pause" );
   return 0;
}
```

Random Number Generator (Mid Square Method):

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

unsigned long long int randm(int n);
unsigned long long int von(unsigned long long int x, int n);

int main(void){
```

```
   unsigned long long int x, s;
   int n, i, r;
   printf("Enter the number of digits in the seed value ");
   scanf("%d", &n);

   printf("\nEnter the total number of random numbers to be generated ");
   scanf("%d", &r);

   if (n >= 12){
   printf("TOO LARGE!!");
   exit(0);
   }

   x = randm(n);
   for(i = 0; i < r; i++){
      s = von(x, n);
      x = s;
      printf("\nRandom Number generated: %lld\n", s);
   }

 return 0;

}



unsigned long long int randm(int n)

{

 double x;

 unsigned long long int y;

 srand(getpid());

 x = rand() / (double)RAND_MAX;

 y = (unsigned long long int) (x * pow(10.0, n*1.0));

 return y;

}
```

```
unsigned long long int von(unsigned long long int x, int n)

{

 unsigned long long int y;

 int k;

 k = n / 2;

 y =(unsigned long long int)((x / pow(10.0, k * 1.0)) * x) % (unsigned long long int)
(pow(10.0, n * 1.0));

 return y;

}
```

User Interface for Random Number Generator:

```
Form1.Designer.cs
namespace WindowsFormsApplication2
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
      protected override void Dispose(bool disposing)
```

```
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.textBox7 = new System.Windows.Forms.TextBox();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.button4 = new System.Windows.Forms.Button();
            this.label3 = new System.Windows.Forms.Label();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.SystemColors.ControlDark;
            this.button1.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button1.Location = new System.Drawing.Point(635, 195);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(182, 47);
            this.button1.TabIndex = 0;
            this.button1.Text = "Generate";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.SystemColors.ControlDark;
```

```
        this.button2.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.button2.Location = new System.Drawing.Point(635, 258);
        this.button2.Name = "button2";
        this.button2.Size = new System.Drawing.Size(182, 47);
        this.button2.TabIndex = 1;
        this.button2.Text = "Connect";
        this.button2.UseVisualStyleBackColor = false;
        this.button2.Click += new System.EventHandler(this.button2_Click);
        //
        // button3
        //
        this.button3.BackColor = System.Drawing.SystemColors.ControlDark;
        this.button3.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.button3.Location = new System.Drawing.Point(633, 477);
        this.button3.Name = "button3";
        this.button3.Size = new System.Drawing.Size(184, 47);
        this.button3.TabIndex = 2;
        this.button3.Text = "Database";
        this.button3.UseVisualStyleBackColor = false;
        this.button3.Click += new System.EventHandler(this.button3_Click);
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.label1.Location = new System.Drawing.Point(673, 12);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(115, 29);
        this.label1.TabIndex = 3;
        this.label1.Text = "Algorithm";
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.label2.Location = new System.Drawing.Point(673, 102);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(105, 29);
        this.label2.TabIndex = 4;
        this.label2.Text = "Numbers";
        //
```

```
            // textBox4
            //
            this.textBox4.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.textBox4.Location = new System.Drawing.Point(12, 12);
            this.textBox4.Multiline = true;
            this.textBox4.Name = "textBox4";
            this.textBox4.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.textBox4.Size = new System.Drawing.Size(602, 512);
            this.textBox4.TabIndex = 13;
            this.textBox4.TextChanged += new
System.EventHandler(this.textBox4_TextChanged);
            //
            // textBox7
            //
            this.textBox7.Location = new System.Drawing.Point(635, 145);
            this.textBox7.Multiline = true;
            this.textBox7.Name = "textBox7";
            this.textBox7.Size = new System.Drawing.Size(182, 33);
            this.textBox7.TabIndex = 16;
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(635, 366);
            this.textBox1.Multiline = true;
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(182, 33);
            this.textBox1.TabIndex = 17;
            this.textBox1.TextChanged += new
System.EventHandler(this.textBox1_TextChanged_1);
            //
            // button4
            //
            this.button4.BackColor = System.Drawing.SystemColors.ControlDark;
            this.button4.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button4.Location = new System.Drawing.Point(633, 415);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(184, 47);
            this.button4.TabIndex = 18;
            this.button4.Text = "Show Session";
            this.button4.UseVisualStyleBackColor = false;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // label3
            //
```

```
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Corbel", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.Location = new System.Drawing.Point(688, 324);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(90, 29);
            this.label3.TabIndex = 19;
            this.label3.Text = "Session";
            //
            // comboBox1
            //
            this.comboBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "BBS",
            "MSM",
            "LCG",
            "LFG"});
            this.comboBox1.Location = new System.Drawing.Point(635, 53);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(182, 33);
            this.comboBox1.TabIndex = 20;
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.SystemColors.ControlLight;
            this.ClientSize = new System.Drawing.Size(836, 544);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.textBox7);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.ForeColor = System.Drawing.SystemColors.ActiveCaptionText;
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
```

```
        this.ImeMode = System.Windows.Forms.ImeMode.NoControl;
        this.Name = "Form1";
        this.Text = "Random Number Generator";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.TextBox textBox4;
    private System.Windows.Forms.TextBox textBox7;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Button button4;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.ComboBox comboBox1;
  }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Runtime.InteropServices;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace WindowsFormsApplication2
{
```

```
public partial class Form1 : Form
{
    int isConnected = 0;
    int SSID;
    SqlConnection cnn;
    SqlCommand com;
    SqlDataReader dout;

    [DllImport("BlumBlumShub.dll")]
    public static extern IntPtr bbs(int l);
    [DllImport("ConsoleApplication3.dll")]
    public static extern IntPtr random_n(int l, int m);
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {

    }

    private void listBox2_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void checkedListBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
```

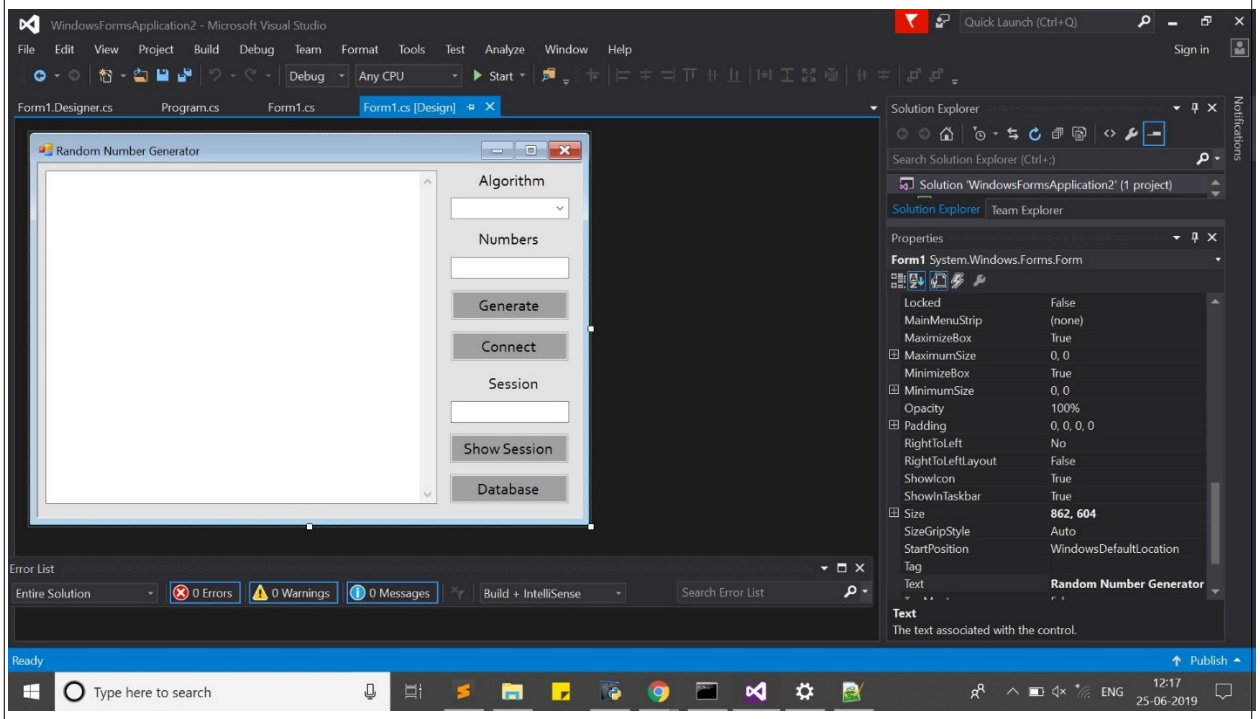
```
    }

    private void button1_Click(object sender, EventArgs e)
    {
        int m;
        string METHOD = comboBox1.Text;
        string MET = comboBox1.Text;
        if (comboBox1.Text == "LCG")
        {
            METHOD = "Linear Congruence Generator";
            m = 0;
        }
        else if (comboBox1.Text == "LFG")
        {
            METHOD = "Linear Fibonacci Generator";
            m = 1;
        }
        else if (comboBox1.Text == "MSM")
        {
            METHOD = "Middle Square Method";
            m = 2;
        }
        else
        {
            METHOD = "Blum Blum Shub";
            m = 3;
        }
        SSID = SSID + 1;
        string NUMBERS = textBox7.Text;
        int n = Int32.Parse(textBox7.Text);
        //IntPtr ptr = bbs(n);
        IntPtr ptr = random_n(n, m);
        IntPtr start = IntPtr.Add(ptr, 4);
        int[] result = new int[n];
        Marshal.Copy(start, result, 0, n);
        string s = "Method used : " + METHOD + Environment.NewLine + "Numbers
generated : " + NUMBERS + Environment.NewLine + "Session Number : " +
SSID.ToString() + Environment.NewLine;
        int i;
        for(i = 0;i < n; i++)
        {
            uint k = (uint)result[i];
            string nstring = k.ToString();
            s = s + nstring + Environment.NewLine;
            if(isConnected == 1)
```

```
            {

                string sqlinsertquery = "insert into NumbersTable ( SSID, MET, NUM )
values (" + SSID.ToString() + ",'" + MET + "'," + nstring + ")";
                com = new SqlCommand(sqlinsertquery, cnn);
                com.ExecuteNonQuery();
            }
        }
        textBox4.Text = s;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        string connetionString;
        connetionString = @"Data Source=FINIQ738\SQL_2016;Initial
Catalog=NumbersDatabase;User ID=sa;Password=Password!23";
        cnn = new SqlConnection(connetionString);
        cnn.Open();
        textBox4.Text = "Connection Established";
        isConnected = 1;
        string sqlsession = "select Max(SSID) from NumbersTable";
        com = new SqlCommand(sqlsession, cnn);
        dout = com.ExecuteReader();
        dout.Read();
        if(dout.GetValue(0) != DBNull.Value)
            SSID = (int)dout.GetValue(0);
        dout.Close();
        textBox4.Text = textBox4.Text + Environment.NewLine + "Last Session : " +
SSID.ToString();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (isConnected == 0)
            textBox4.Text = "Database not connected" + Environment.NewLine;
        else
        {
            string sqlshowquery = "select * from NumbersTable";
            com = new SqlCommand(sqlshowquery, cnn);
            dout = com.ExecuteReader();
            string numbers = "";
            while (dout.Read())
            {
                numbers = numbers + dout.GetValue(2) + Environment.NewLine;
            }
```

```
            dout.Close();
            textBox4.Text = numbers;
        }
    }

    private void textBox1_TextChanged_1(object sender, EventArgs e)
    {

    }

    private void button4_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "")
            textBox4.Text = "Please enter Session Number";
        else
        {
            if (isConnected == 0)
                textBox4.Text = "Database not connected" + Environment.NewLine;
            else
            {
                string sqlshowquery = "select * from NumbersTable where SSID=" +
textBox1.Text;
                com = new SqlCommand(sqlshowquery, cnn);
                dout = com.ExecuteReader();
                string numbers = "";
                string method_show = "BBS";
                int count = 0;
                while (dout.Read())
                {
                    count = count + 1;
                    method_show = dout.GetValue(1).ToString();
                    numbers = numbers + dout.GetValue(2) + Environment.NewLine;
                }
                dout.Close();
                string METstring;
                if (method_show == "LCG")
                {
                    METstring = "Linear Congruence Generator";
                }
                else if (method_show == "LFG")
                {
                    METstring = "Linear Fibonacci Generator";
                }
                else if (method_show == "MSM")
                {
```

```
                METstring = "Middle Square Method";
            }
            else
            {
                METstring = "Blum Blum Shub";
            }
            textBox4.Text = "Showing Session : " + textBox1.Text +
Environment.NewLine + "Method used : " + METstring + Environment.NewLine +
"Numbers Generated : " + count.ToString() + Environment.NewLine + numbers;
        }
      }
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }
  }
}

Program.cs
using System;
using System.Runtime.InteropServices;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
  static class Program
  {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>



    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
    }
  }
```

```
}
```



WEEK 2

The second assignment consists mainly of the following parts:

1. Newton Raphson Method coding (Done)
2. Option pricing via binomial tree
3. Random Number generator showing error with respect to number of simulations

4. Cholesky Decomposition for large matrices (Done)
5. Program to input mean and standard deviation and generate numbers with less error and more count
6. Cubic spline least square
7. linear regression fit for 10 numbers where 1 number is not provided

Cholesky's Decomposition:

```
import math
def cholesky(A, n):
    L = [[0.0] * len(A) for _ in range(len(A))]
    for i, (Ai, Li) in enumerate(zip(A, L)):
        for j, Lj in enumerate(L[:i+1]):
            s = sum(Li[k] * Lj[k] for k in range(j))
            Li[j] = math.sqrt(Ai[i] - s) if (i == j) else \
                    (1.0 / Lj[j] * (Ai[j] - s))
    #Lower Triangular Matrix is to be printed
    print('Lower Triangular Matrix:')
    for i in range(n):
        print('[', end = '  ')
        for j in range(n):
            print('%.4f'%L[i][j], end='  ')
        print('', end = ']\n')
    #the transpose of the matrix
    print('Transpose Matrix:')
    for i in range(n):
        print('[', end = '  ')
        for j in range(n):
            print('%.4f'%L[j][i], end='  ')
        print('', end = ']\n')

n = 4
#Hardcoding the Matrix
matrix = [[18, 22, 54, 42],
[22, 70, 86, 62],
[54, 86, 174, 134], [42, 62, 134, 106]]
#call the cholesky function
cholesky(matrix, n)
```

Cubic Spline Method:

```
from patsy import dmatrix
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


#read the data from the CS File
data = pd.read_csv("D:\Assignment2\wage.csv")

data.head()

data_x = data['age']
data_y = data['wage']

#dividing data into train and validation sets
from sklearn.model_selection import train_test_split
train_x, valid_x, train_y, valid_y = train_test_split(data_x, data_y, test_size = 0.33,
random_state = 1)


#Generating cubic spline with 3 knots at 24, 40 and 60

transformed_x = dmatrix("bs(train, knots=(24, 40 ,60), degree=3,
include_intercept=False)", {"train": train_x}, return_type = 'dataframe')

fit1 = sm.GLM(train_y, transformed_x).fit()

#Generating cubic spline with 3 knots at 24, 40 and 60

transformed_x2 = dmatrix("bs(train, knots=(24, 40 ,60), degree=3,
include_intercept=False)", {"train": train_x}, return_type = 'dataframe')

fit2 = sm.GLM(train_y, transformed_x2).fit()

# Predictions on both splines
pred1 = fit1.predict(dmatrix("bs(valid, knots=(25,40,60), include_intercept=False)",
{"valid": valid_x}, return_type='dataframe'))
pred2 = fit2.predict(dmatrix("bs(valid, knots=(25,40,50,65),degree =3,
include_intercept=False)", {"valid": valid_x}, return_type='dataframe'))

# Calculating RMSE values
rms1 = math.sqrt(mean_squared_error(valid_y, pred1))
print(rms1)
rms2 = math.sqrt(mean_squared_error(valid_y, pred2))
print(rms2)
```

```
# We will plot the graph for 70 observations only
xp = np.linspace(valid_x.min(),valid_x.max(),70)

# Make some predictions
pred1 = fit1.predict(dmatrix("bs(xp, knots=(25,40,60), include_intercept=False)",
{"xp": xp}, return_type='dataframe'))
pred2 = fit2.predict(dmatrix("bs(xp, knots=(25,40,50,65),degree =3,
include_intercept=False)", {"xp": xp}, return_type='dataframe'))

# Plot the splines and error bands
plt.scatter(data.age, data.wage, facecolor='None', edgecolor='k', alpha=0.1)
plt.plot(xp, pred1, label='Specifying degree =3 with 3 knots')
plt.plot(xp, pred2, color='r', label='Specifying degree =3 with 4 knots')
plt.legend()
plt.xlim(15,85)
plt.ylim(0,350)
plt.xlabel('age')
plt.ylabel('wage')
plt.show()
```

Day from Date:

```
#Hardcoded the Days of the Week
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
#Take Date as Input
g = input("Enter date in dd/mm/yy format: ")
#splitting the date
a = g.split('/')
day = int(a[0])
year = int(a[2])
month = int(a[1])
```

```
def day_of_week(day, month, year):
    t = [0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4]
    year -= month < 3
    i = (year + int(year/4) - int(year/100) + int(year/400) + t[month-1] + day) % 7
    return days[i]
#call the week
print(day_of_week(day, month, year))
```

Linear Regression Fit:

```
#import modules
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt


#read the data from the CS File
data = pd.read_csv("D:\Assignment2\wage.csv")

data.head()

data_x = data['age']
data_y = data['wage']

#dividing data into train and validation sets
from sklearn.model_selection import train_test_split
train_x, valid_x, train_y, valid_y = train_test_split(data_x, data_y, test_size = 0.33,
random_state = 1)

#visualise the relationship between age and wage
#plt.scatter(train_x, train_y, facecolor = 'None', edgecolor = 'k', alpha = 0.3)
#plt.show()

from sklearn.linear_model import LinearRegression
#fitting linear regression model
x = train_x.values.reshape(-1, 1)
model = LinearRegression()
model.fit(x, train_y)
#print(model.coef_)
#print(model.intercept_)
```

```
#prediction on validation dataset

valid_x = valid_x.values.reshape(-1, 1)
pred = model.predict(valid_x)

#visualisation

xp = np.linspace(valid_x.min(), valid_x.max(), 70)
xp = xp.reshape(-1, 1)
pred_plot = model.predict(xp)

plt.scatter(valid_x, valid_y, facecolor = 'None', edgecolor='k', alpha = 0.3)
plt.plot(xp, pred_plot)
plt.show()
```

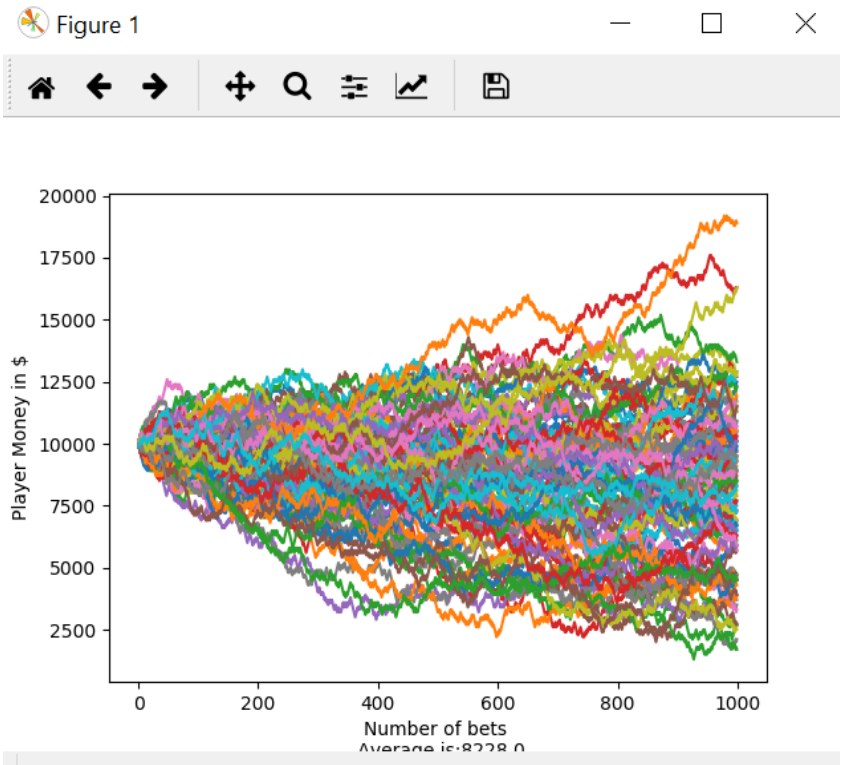
WEEK 3

Monte Carlo Analysis for Dice Game:

```
#Import libraries
import random
import matplotlib.pyplot as plt
#Create function for simulating die roll
#The die can take values from 1 to 100. If the number is between 1 #and 51, the house wins.
#If the number is between 52 and 100, the player wins.
def rolldice():
```

```
    dice = random.randint(1,100)

    if dice <=51:
        return False
    elif dice >51 & dice <=100:
        return True

#Define a function for the play which takes 3 arguments :
#1. total_funds = total money in hand the player is starting with
#2. wager_amount = the betting amount each time the player plays
#3. total_plays = the number of times the player bets on this game
def play(total_funds, wager_amount, total_plays):

    #Create empty lists for :
    # 1.Play_number and
    # 2.Funds available
    # 3.Final Fund
    Play_num = []
    Funds = []
#Start with play number 1
    play = 1
#If number of plays is less than the max number of plays we have set
    while play < total_plays:
        #If we win
        if rolldice():
            #Add the money to our funds
            total_funds = total_funds + wager_amount
            #Append the play number
            Play_num.append(play)
            #Append the new fund amount
            Funds.append(total_funds)
        #If the house wins
        else:
            #Add the money to our funds
            total_funds = total_funds - wager_amount
            #Append the play number
            Play_num.append(play)
            #Append the new fund amount
            Funds.append(total_funds)

        #Increase the play number by 1
        play = play + 1

    #Line plot of funds over time
```

```
    plt.plot(Play_num,Funds)
    Final_funds.append(Funds[-1])
    return(Final_funds)

#Call the function to simulate the plays and calculate the remaining #funds of the
player after all the bets
#Intialize the scenario number to 1
x=1
#Create a list for calculating final funds
Final_funds= []
while x<=100:
    ending_fund = play(10000,100,1000)
    x=x+1
#Plot the line plot of "Account Value" vs "The number of plays"
plt.ylabel('Player Money in $')
plt.xlabel("Number of bets \n Average is:" + str(sum(ending_fund)/len(ending_fund)))
plt.show()
#Print the money the player ends with
print("The player starts the game with $10,000 and ends with $" +
str(sum(ending_fund)/len(ending_fund)))
ending_fund = play(10000,100,5)
```

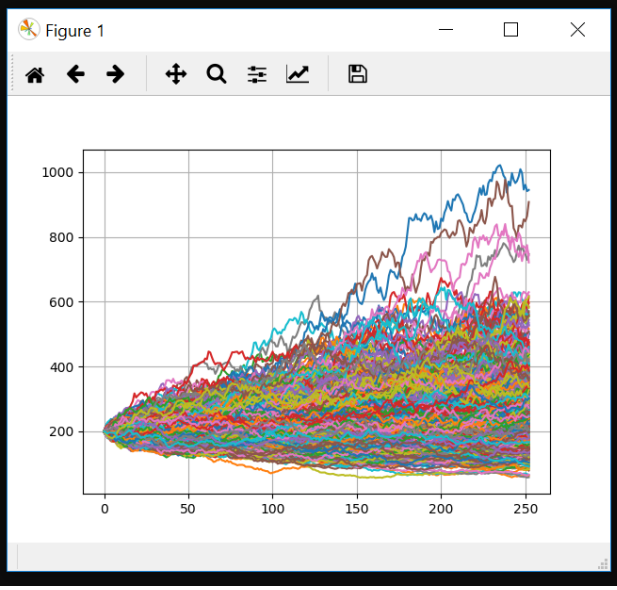


Monte Carlo Analysis of Volatility

```
#import necessary packages
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.stats import norm
from pandas_datareader import data

#download Apple price data into DataFrame
apple = data.DataReader('AAPL', 'yahoo',start='1/1/2000')

#calculate the compound annual growth rate (CAGR) which
#will give us our mean return input (mu)
days = (apple.index[-1] - apple.index[0]).days
cagr = ((((apple['Adj Close'][-1]) / apple['Adj Close'][1])) ** (365.0/days)) - 1
print ('CAGR =',str(round(cagr,4)*100)+"%")
mu = cagr

#create a series of percentage returns and calculate
#the annual volatility of returns
```

```
apple['Returns'] = apple['Adj Close'].pct_change()
vol = apple['Returns'].std()*math.sqrt(252)
print ("Annual Volatility =",str(round(vol,4)*100)+"%")
#Define Variables
S = apple['Adj Close'][-1] #starting stock price (i.e. last available real stock price)
T = 252 #Number of trading days
mu = round(cagr,4)  #Return
vol = round(vol,4) #Volatility
for i in range(1000):
    #create list of daily returns using random normal distribution
    daily_returns=np.random.normal(mu/T,vol/math.sqrt(T),T)+1
    #set starting price and create price series generated by above random daily returns
    price_list = [S]

    for x in daily_returns:
        price_list.append(price_list[-1]*x)

    #plot data from each individual run which we will plot at the end
    plt.plot(price_list)
print(max(daily_returns))

print(min(daily_returns))
#show the plot of multiple price series created above
plt.grid()
plt.show()
```

```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\FinIQ>python.exe D:\Assignment1\monteCarloVolatility.py
CAGR = 23.05%
Annual Volatility = 40.72%
1.0833338171007967
0.930272918914206
```



Option Pricing Using Binomial Trees:

```
#----------------------------------------Code For
Plotting-----------------------------------------------------
import matplotlib.pyplot as plt
import networkx as nx
```

```
def display_binomial_model(N, net_stock, net_option):
    G=nx.DiGraph()
    num = 1
    x = 1
    a = (N/5)+0.5
    #print(a)
    plt.rcParams['figure.dpi'] = np.maximum(72,(72*a))
    #print(plt.rcParams['figure.dpi'])

    for i in range(N+1):
        for j in range(i+1):
            G.add_node(num, pos=(2*a*i,(4*a*j-2*a*i)),
                       stock=round(net_stock[j,i],4), option=round(net_option[j,i],4))
            num = num+1
            if(j>=1):
                G.add_edge(x,(x+i))
                G.add_edge(x,(x+i+1))
                x=x+1

    labels1 = {}
    for num, temp in nx.get_node_attributes(G, 'stock').items():
        labels1.setdefault(num,"")
        labels1[num]+=str(temp)
    for num, temp in nx.get_node_attributes(G, 'option').items():
        labels1.setdefault(num,"")
        labels1[num]+="\n"+str(temp)

    pos1=nx.get_node_attributes(G, 'pos')
    #nx.draw(G, pos=nx.circular_layout(G), labels=nx.get_node_attributes(G, 'stock'),
with_labels=True)
    nx.draw(G, pos=pos1, labels=labels1, with_labels=True,
node_size=np.minimum(2500,(2500/(0.9*a))), node_color="skyblue", node_shape="o",
alpha=0.9,
            linewidths=4, font_size=np.minimum(12,(12/(0.75*a))), font_color="grey",
font_weight="bold", width=2, edge_color="lightgrey")
    #print(np.maximum(100,(150*a)))
    plt.show()


##----------------------------------------MAIN
CODE-----------------------------------------------------------------
#Option pricing using binomial tree
import numpy as np
import math
import matplotlib.pyplot as plt
```

```
import networkx as nx

def binomial_model(N, T, S0, K, sigma, r, c1, c2):
    """
    N = number of binomial iterations/time steps
    T = expiry time in years
    S0 = initial stock price
    K = strike price
    sigma = volatility of asset
    r = risk free interest rate per annum

    dt = length of time step
    u = factor change of upstate
    d = factor change of downstate
    """
    dt = T / N
    #u = math.exp(sigma*math.sqrt(dt))
    #print(u)
    #d = 1 / u
    u = 1.2
    d = 0.8
    p = (math.exp(r*dt) - d) / (u - d)

    # make stock price tree
    stock = np.zeros([N + 1, N + 1])
    for i in range(N + 1):
        for j in range(i + 1):
            stock[j, i] = S0 * (u ** (i - j)) * (d ** j)
    #print(stock)

    # Generate option prices recursively
    option = np.zeros([N + 1, N + 1])
    if (c1 == "C"):
        option[:, :] = np.maximum(np.zeros(N + 1), np.triu((stock[:, :] - K), k=0)) #call
option
    elif (c1 == "P"):
        option[:, :] = np.maximum(np.zeros(N + 1), np.triu((K - stock[:, :]), k=0)) #put
option

    if (c2 == "E"):
        for i in range(N - 1, -1, -1):
            for j in range(0, i + 1):
                option[j, i] = math.exp((-1)*r*dt) * (p*option[j, i + 1] + (1-p)*option[j + 1, i +
1])
```

```
    elif (c2 == "A"):
        for i in range(N - 1, -1, -1):
            for j in range(0, i + 1):
                temp = math.exp((-1)*r*dt)*(p*option[j, i + 1] + (1-p)*option[j + 1, i + 1])
                option[j, i] = np.maximum(temp,(option[j, i]))

    #print(option)
    return stock,option


if __name__ == "__main__":
    print("Option price by Binomial Tree Model:")
    #binomial_model(N, T, S0, K, sigma, r, c1, c2)
    #op_price = binomial_model(no_iterations, expiry_time, initial, strike, volatility,
interest, choice_call_put, choice_eu_amer)
    no_iterations=2
    net_stock, net_option = binomial_model(no_iterations, 2, 50, 52, 0.3, 0.05, "P", "E")
    display_binomial_model(no_iterations, net_stock, net_option)
```

```
Option price by Binomial Tree Model:
```