FUNCTION MAXIMA EVALUATION
USING GENETIC ALGORITHM

```
import java.io.*;
import java.lang.Math;
import java.util.*;
import javax.script.*;

class GA4
 {
   static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
   static ScriptEngine engine = (new ScriptEngineManager()).getEngineByName("JavaScript");
   static int nvar=0,len=0,maxvar=10;
   static double x[][]=new double[maxvar][5]; //FIELDS: en preci lwrLim uprLim length
   static double n[];
   public static void main(String args[]) throws Exception
    {
      System.out.print("\nFunction\t : ");
      String fneq=br.readLine();
      //EXAMPLE : fneq=21.5+x1*Math.sin(4*Math.PI*x1)+x2*Math.sin(20*Math.PI*x2);
      int i,j,gen;
      for(i=0;i<10;i++)
         x[i][0]=0;
      final int len=fnvr(fneq);
      System.out.print("Population Size\t : ");
      final int pop_size=Integer.parseInt(br.readLine());
      System.out.print("Crossover Prob.\t : ");
      final double pc=Double.parseDouble(br.readLine());
      System.out.print("Mutation Prob.\t : ");
      final double pm=Double.parseDouble(br.readLine());
      System.out.print("Generations\t : ");
      final int gtot=Integer.parseInt(br.readLine());

      String chr[]=new String [pop_size],tmp;
      //initialize 1st generation
      for(i=0;i<pop_size;i++)
       {
         tmp="";
         for(j=0;j<len;j++)
          {
            if(Math.random()>0.5) tmp+="1";
            else tmp+="0";
          }
         chr[i]=tmp;
       }
      System.out.println("\nSTART OF GENETIC ITERATION");
      int eop,pos,c1=0,c2=0;
      double r,ns,best[]=new double [nvar+2];
double pqr[]=new double [pop_size];
n=new double[nvar];
String chrt[]=new String [pop_size]; //temp Str arr : next gen chromosomes
//main looping for each generation
for(gen=0;gen<gtot;gen++)
 {
    System.out.println("Generation : "+(gen+1));
    for(i=0;i<pop_size;i++)
     {
        pos=0;
        for(j=0;j<nvar;j++)
         {
            n[j]=B2D(chr[i].substring(pos,pos+(int)x[j][4]),j);
            pos+=(int)x[j][4];
         }
        if(i==0)
         {
            pqr[i]=evalt(fneq);
            if(pqr[i]>best[0])
             {
                best[0]=pqr[i];
                best[1]=gen+1;
                for(j=0;j<nvar;j++)
                    best[j+2]=n[j];
             }
         }
        else
         {
            pqr[i]=pqr[i-1]+evalt(fneq);
            if((pqr[i]-pqr[i-1])>best[0])
             {
                best[0]=pqr[i]-pqr[i-1];
                best[1]=gen+1;
                for(j=0;j<nvar;j++)
                    best[j+2]=n[j];
             }
         }
     }
    System.out.println("Total Value = "+pqr[pop_size-1]);
    //select chromosomes for next generation
    for(i=0;i<pop_size;i++)
        pqr[i]/=pqr[pop_size-1]; //cumulative prob. Roulette Wheel
    for(i=0;i<pop_size;i++)
     {
        r=Math.random();
        for(j=0;j<pop_size;j++)
         {
            if(r<pqr[j])
             {
                chrt[i]=chr[j];
            break;
         }
     }
 }
//Crossover
ns=0;
for(i=0;i<pop_size;i++)
 {
    if(Math.random()<pc)
     {
        pqr[i]=1;
        ns++;
     }
    else pqr[i]=0;
 }
if(ns%2!=0) //odd no. of selected chromosome selected for pairing
    ns=Math.random()*(pop_size-ns);
for(i=0;i<pop_size;i++) //select extra chromosome for crossover pairing
 {
    if(pqr[i]==0) ns--;
    if(ns==0)
     {
        pqr[i]=1;
        break;
     }
 }
//perform Crossover
for(i=0;i<pop_size;i++)
 {
    if(pqr[i]==0) continue;
    eop=0;
    if(eop==0)
     {
        c1=i;
        eop=1;
     }
    else if(eop==1)
     {
        c2=i;
        eop=2;
     }
    if(eop==2)
     {
        pos=(int)Math.random()*len;
        chrt[c1]=chrt[c1].substring(0,pos)+chrt[c2].substring(pos);
        chrt[c2]=chrt[c2].substring(0,pos)+chrt[c1].substring(pos);
        eop=0;
     }
 }
//Mutation
```

```
      for(i=0;i<pop_size*len;i++)
       {
         if(Math.random()<pm)
          {
            j=i/len;
            pos=i%len;
            if(chrt[j].charAt(pos)=='0')
                chrt[j]=chrt[j].substring(0,pos)+"1"+chrt[j].substring(pos);
            else
                chrt[j]=chrt[j].substring(0,pos)+"0"+chrt[j].substring(pos);
          }
       }
      for(i=0;i<pop_size;i++)
         chr[i]=chrt[i];
    }
   //Display Best Values
   System.out.println("\nBest Value : "+best[0]);
   System.out.println("Generation : "+best[1]);
   System.out.println("Values     : ");
   for(i=2;i<nvar+2;i++)
      System.out.println("x"+(i-1)+" \t: "+best[i]);
 }
//Method to evaluate string expression using JavaScript
public static double evalt(String fneq)throws Exception
 {
   double fn=0;
   for(int i=0;i<nvar;i++)
    {
      engine.put(("x"+(i+1)),n[i]);
     }
   fn=Double.parseDouble(engine.eval(fneq)+"");
   return fn;
 }
//Method to convert binary chromosome sequence to its numerical value
public static double B2D(String s, int vn)
 {
   double n=0.0;
   for(int i=0;i<s.length();i++)
      if(s.charAt(i)=='1') n+=Math.pow(2,(s.length()-i-1));
   n=x[vn][2]+n*(x[vn][3]-x[vn][2])/(Math.pow(2,s.length())-1);
   return n;
 }
//Method to evaluate expression variables and their parameters
public static int fnvr(String fneq)throws Exception
 {
   int i,j,len=0;
   for(i=0;i<fneq.length();i++)
    {
      if(fneq.charAt(i)=='x')
       {
```

```
                int idx=Integer.parseInt(fneq.charAt(i+1)+"");
                idx--;
                if(x[idx][0]==0)
                 {
                    System.out.println("Variable\t : "+(idx+1));
                    System.out.print("Precision\t : ");
                    x[idx][1]=Double.parseDouble(br.readLine());
                    System.out.print("Lower Limit\t : ");
                    x[idx][2]=Double.parseDouble(br.readLine());
                    System.out.print("Upper Limit\t : ");
                    x[idx][3]=Double.parseDouble(br.readLine());
                    x[idx][0]=1;
                    nvar++;
                    for(j=1;;j++)
                     {
                        if(Math.pow(2,j)>(x[idx][1]*(x[idx][3]-x[idx][2])))
                         {
                            x[idx][4]=j;
                            break;
                         }
                     }
                 }
             }
         }
        for(i=0;i<nvar;i++)
            len+=x[i][4];
        return (len);
    }
}
```

TERMINAL WINDOW :

```
Function                    : 21.5+x1*Math.sin(4*Math.PI*x1)+x2*Math.sin(20*Math.PI*x2)
Variable                    : 1
Precision                   : 10000
Lower Limit                 : -3.0
Upper Limit                 : 12.1
Variable                    : 2
Precision                   : 10000
Lower Limit                 : 4.1
Upper Limit                 : 5.8
Population Size             : 20
Crossover Prob.             : 0.25
Mutation Prob.              : 0.01
Generations                 : 10000
```

```
START OF GENETIC ITERATION
Generation : 1
Total Value = 409.79377257070024
Generation : 2
Total Value = 445.9614442942778
Generation : 3
Total Value = 459.3753003342473
Generation : 4
Total Value = 448.226443550758
Generation : 5
Total Value = 473.81885531853123
Generation : 6
Total Value = 509.41613116246873
.
.
.
Generation : 6914
Total Value = 598.3270407702545
Generation : 6915
Total Value = 598.6102671938445
Generation : 6916
Total Value = 579.8202426974135
Generation : 6917
Total Value = 603.6994231395291
Generation : 6918
Total Value = 601.9279309504764
Generation : 6919
Total Value = 606.0494566642751
Generation : 6920
Total Value = 545.9536540717735
.
.
.
Generation : 9997
Total Value = 605.2871812340687
Generation : 9998
Total Value = 582.4194264851458
Generation : 9999
Total Value = 610.8088485016723
Generation : 10000
Total Value = 585.362350731463

Best Value        : 38.84702806965913
Generation        : 6917.0
Values            :
x1                : 11.623918243096325
x2                : 5.724771874141667
```