Java Homework 2

1. Command window input and output practicing.

Write an application program that inputs from the user the radius of a circle as an integer and prints the circle’s diameter, circumference and area.

(Please use the command window input and output)

*Diameter =* 2 r

*Circumference=* 2πr

*Area =* π**

You may use the predefined const Math.PI for the value of π

Note that the values produced by the circumference and area are floating-point numbers.

2. (a).Please execute, observe, **resize the window** and compare the following four cases of the java application programs and **write the differences among these four cases (from the syntax and from the result points).**

(b). Please convert the (a) java application program into java applet program:

(You may extend this program into any sort algorithm)

//**Case I: SortFrame.java**

import java.awt.\*;

import javax.swing.\*;

import java.awt.Color;

public class SortFrame extends JFrame

{ int numbers[ ] = **new int [100]**;

public SortFrame( )

{

super("SortJFrame");

setSize(500,600)

setVisible(true); /**/automatically call paint()method**

}

public void paint(Graphics g)

{ int i,j;

super.paint(g);

g.setColor(Color.magenta);

for(j = 0; j < numbers.length; ++j)

numbers[j] = **(int)(Math.random()\*101)**;

for(j = 0; j < numbers.length; ++j)

g.drawLine(10, 20+3\*j, 10 + numbers[j], 20+3\*j);

g.drawString("The original array", 10, 350);

bubblesort();

for(j = 0; j < numbers.length; ++j)

g.drawLine( 225, 20+3\*j, 225 + numbers[j], 20+3\*j);

g.drawString("The sorted array", 225,350); }

public void bubblesort()

{ int size = numbers.length;

int x, y, temp;

for(x = 0; x < size – 1; ++x)

{ for( y = 0; y < size – 1 – x; ++y)

if(numbers[y] > numbers[y +1] )

{ temp = numbers[y];

numbers[y] = numbers[y+1];

numbers[y+1] = temp; } }

}

public static void main(String args[ ])

{

SortFrame  **f1 = new SortFrame( );**

f1.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE ); }

}

**//CaseII SortFrame2.java**

import java.awt.\*;

import javax.swing.\*;

import java.awt.Color;

public class SortFrame2 extends JFrame

{ int numbers[ ] = new int [100];

public SortFrame2( )

{

super("SortJFrame");

for(int j = 0; j < numbers.length; ++j)

numbers[j] = (int)(Math.random()\*101);

setSize(500,600);

setVisible(true); **//automatically call paint()method** }

public void paint(Graphics g)

{ int i,j;

super.paint(g);

g.setColor(Color.magenta);

for(j = 0; j < numbers.length; ++j)

g.drawLine(10, 20+3\*j, 10 + numbers[j], 20+3\*j);

g.drawString("The original array", 10, 350);

bubblesort();

for(j = 0; j < numbers.length; ++j)

g.drawLine(225, 20+3\*j, 225 + numbers[j], 20+3\*j);

g.drawString("The sorted array", 225,350); }

public void bubblesort() //this bubble sort routine sorts integers from low to high

{ int size = numbers.length;

int x, y, temp;

for(x = 0; x < size – 1; ++x)

{ for( y = 0; y < size – 1 – x; ++y)

if(numbers[y] > numbers[y +1] )

{ temp = numbers[y];

numbers[y] = numbers[y+1];

numbers[y+1] = temp; } } }

public static void main(String args[ ])

{

SortFrame2 f1 = new SortFrame2( );

f1.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE ); }

}

**//Case III SortPanel101.java**

import java.awt.\*;

import javax.swing.\*;

import java.awt.Color;

public class SortPanel101 extends JPanel

{ int number1[ ] = new int [100];

int number2[ ] = new int [100];

public SortPanel101( )

{

for(int j = 0; j < number1.length; ++j)

{ number1[j] = (int)(Math.random()\*101);

number2[j]=number1[j]; }

setSize(500,600);

setVisible(true); }

public void paintComponent(Graphics g)

{ int i,j;

super.paintComponent(g);

g.setColor(Color.magenta);

for(j = 0; j < number1.length; ++j)

g.drawLine( 10, 10+3\*j, 10 + number1[j], 10+3\*j );

g.drawString("The original array", 10, 325);

System.out.println(" Original");

bubblesort();

for(j = 0; j < number2.length; ++j)

g.drawLine(225, 10+3\*j, 225 + number2[j], 10+3\*j);

g.drawString("The sorted array", 225,325);

System.out.println(" After sort");

}

public void bubblesort()

{ //we can obtain the size of a Java array like this:

int size = number2.length;

int x, y, temp;

for(x = 0; x < size – 1; ++x)

{ for( y = 0; y < size – 1 – x; ++y)

if(number2[y] > number2[y +1] )

{ temp = number2[y];

number2[y] = number2[y+1];

number2[y+1] = temp; }

}

}

public static void main(String args[ ])

{

**SortPanel101 p = new SortPanel101( );**

**JFrame w = new JFrame();**

w.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

**w.add(p);**

**w.setSize(500,600);**

**w.setVisible(true);**

}

}

**// Case IV: contains SortPanel.java and DrawSortTest.java--- TWO files**

//File: SortPanel.java

import java.awt.\*;

import javax.swing.\*;

import java.awt.Color;

public class SortPanel extends JPanel

{ int numbers[ ] = new int [100];

public void paintComponent(Graphics g)

{ int i,j;

super.paintComponent(g);

g.setColor(Color.magenta);

for(j = 0; j < numbers.length; ++j)

numbers[j] = (int)(Math.random()\*101);

for(j = 0; j < numbers.length; ++j)

g.drawLine(10, 20+3\*j, 10 + numbers[j], 20+3\*j);

g.drawString("The original array", 10, 350);

System.out.println(" Original");

bubblesort();

for(j = 0; j < numbers.length; ++j)

g.drawLine(225, 20+3\*j, 225 + numbers[j], 20+3\*j);

g.drawString("The sorted array", 225,350);

System.out.println(" After sort");

}

public void bubblesort()

{ int size = numbers.length;

int x, y, temp;

for(x = 0; x < size – 1; ++x)

{ for( y = 0; y < size – 1 – x; ++y)

if(numbers[y] > numbers[y +1] )

{ temp = numbers[y];

numbers[y] = numbers[y+1];

numbers[y+1] = temp; }

} } }

//File DrawSortTest.java

import javax.swing.JFrame;

public class DrawSortTest

{

public static void main(String args[ ])

{ **SortPanel p = new SortPanel( );**

**JFrame w = new JFrame();**

w.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

**w.add(p);**

w.setSize(500,600);

w.setVisible(true); }

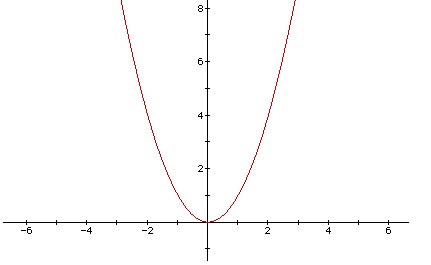
}

3. **Error Curves**

Josephina is a clever girl and addicted to Machine Learning recently. She pays much attention to a method called Linear Discriminant Analysis, which has many interesting properties.

In order to test the algorithm's efficiency, she collects many datasets. What's more, each data is divided into two parts: training data and test data. She gets the parameters of the model on training data and test the model on test data.

To her surprise, she finds each dataset's test error curve is just a parabolic curve. A parabolic curve corresponds to a quadratic function. In mathematics, a quadratic function is a polynomial function of the form *f(x) = ax2 + bx + c*. The quadratic will degrade to linear function if *a = 0*.



It's very easy to calculate the minimal error if there is only one test error curve. However, there are several datasets, which means Josephina will obtain many parabolic curves. Josephina wants to get the tuned parameters that make the best performance on all datasets. So she should take all error curves into account, i.e., she has to deal with many quadric functions and make a new error definition to represent the total error. Now, she focuses on the following new function's minimal which related to multiple quadric functions.

The new function *F(x)* is defined as follow:

*F(x) = max(Si(x))*, *i* = 1...*n*. The domain of *x* is [0, 1000]. *Si(x)* is a quadric function.

Josephina wonders the minimum of *F(x)*. Unfortunately, it's too hard for her to solve this problem. As a super programmer, can you help her?

**Input**

The input contains multiple test cases. The first line is the number of cases *T* (*T* < 100). Each case begins with a number *n*(*n* ≤ 10000). Following *n* lines, each line contains three integers *a* (0 ≤ *a* ≤ 100), *b* (|*b*| ≤ 5000), *c* (|*c*| ≤ 5000), which mean the corresponding coefficients of a quadratic function.

**Output**

For each test case, output the answer in a line. Round to 4 digits after the decimal point.

**Sample Input**

2

1

2 0 0

2

2 0 0

2 -4 2

**Sample Output**

0.0000

0.5000

// Error Curves Using C++ language

// Rujia Liu

#include<stdio.h>

#include<iostream>

#include<algorithm>

using namespace std;

const int maxn = 10000 + 10;

int n, a[maxn], b[maxn], c[maxn];

double F(double x) {

double ans = a[0]\*x\*x+b[0]\*x+c[0];

for(int i = 1; i < n; i++)

ans = max(ans, a[i]\*x\*x+b[i]\*x+c[i]);

return ans;

}

int main() {

int T;

scanf("%d", &T);

while(T--) {

scanf("%d", &n);

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

scanf("%d%d%d", &a[i], &b[i], &c[i]);

double L = 0.0, R = 1000.0;

for(int i = 0; i < 100; i++) {

double m1 = L+(R-L)/3;

double m2 = R-(R-L)/3;

if(F(m1)<F(m2)) R = m2; else L = m1;

}

printf("\nThe Max is %.4lf\n", F(L));

}

system("pause");

return 0;

}

Hint:

n=1, F(x) 是一條拋物線, 怎麼求最小值?

注意到 a>=0, 所以可能退化成一直線或者成為開口向上的拋物線. 後者是標準的下凸函數

, 直線也可視為下凸, 所以n=1時 可視為下凸函數求極值 .

n=2 or n>2 圖形仍然是下凸. 下凸函數求極值可以使用三分法(ternary search) 求解.

具體方法如下:

取區間[L,R]的兩個三分點 m1 and m2. 比較F(m1) and F(m2)大小

Ternary Search:

Let a [unimodal](http://en.wikipedia.org/wiki/Unimodal)(單峰函數) function *f*(*x*) on some interval [*l*; *r*]. Take any two points *m1* and *m2* in this segment:

*l* < *m1* < *m2* < *r*. Then there are three possibilities:

* if *f(m1) < f(m2)*, then the required maximum can not be located on the left side - [*l*; *m1*]. It means that the maximum further makes sense to look only in the interval [*m1*;*r*]
* if *f(m1) > f(m2)*, that the situation is similar to the previous, up to symmetry. Now, the required maximum can not be in the right side - [*m2*; *r*], so go to the segment [*l*; *m2*]
* if *f(m1) = f(m2)*, then the search should be conducted in [*m1*; *m2*], but this case can be attributed to any of the previous two (in order to simplify the code). Sooner or later the length of the segment will be a little less than a predetermined constant, and the process can be stopped.

choice points *m1* and *m2*:

* *m1* = *l* + (*r*-*l*)/3
* *m2* = *r* - (*r*-*l*)/3

Note1: Ternary Search不僅適用在凸函數, 還是用在單峰函數[unimodal](http://en.wikipedia.org/wiki/Unimodal" \o "Unimodal)function. 所謂單峰函數就是先嚴格遞增再嚴格遞減(此時存在唯一的最大值)或者先嚴格遞減再嚴格遞增的函數(此時存在唯一的最大值)