Mike O’Malley’s - Programming Practice Questions

For All Skill Levels. Any programming language.

Lear, Explore, Research, Have Fun !

v0.008, Freeware

Introduction

It is critical that all programmers – students, beginners, intermediate, and even advanced developers with decades of experience who work full time in the industry – get the practice and experience needed to keep their skills improving (and not deteriorating). Programming is like learning to play the piano – it all comes down to practice, practice, practice. If you want a career as a programmer, you need to explore and solve a wide range of topics and keep exploring and solving – it's a life long learning journey. If you don't use it, you lose it !

It is also essential that programmers have reasonable maths skills – so enhance these at every opportunity as well. If math frightens you, then you need to address this. The problems below will help (don't worry, it's only extremely basic maths).

A great thing to do is keep on the lookout for interesting problems or tasks – in movies, TV ads, everyday life - and then write programs to solve / explore these at every opportunity. Keep a list of problems you'd like to solve, and then when you get a quiet morning or evening, and you are feeling like a challenge, look at your list and get cracking !

To get you started, here are some of problems that I really enjoyed solving back when I was getting started with programming (and that I still really enjoy solving now), with various additions and extras. ☺

Notes:

- Do not get bogged down. If you cannot solve something or do something the question asks, move on to another problem and think about the issues and return later to solve it when you can.
- Work together with friends, form a study group with other students in your class, course, or other colleagues at your work, and work together and solve these questions, compare and critique each other's answers, etc. An amazing learning opportunity. ☺

Who this eBook is for

This eBook is for computer programmers – students, beginners, intermediate, and even advanced developers with decades of experience – and it provides a range of interesting problems to explore and solve to help programmers develop, improve and retain skills. And also have fun and learn and have a real sense of achievement when they are solved.

What Programming Language do I need to use ?

The problems in this eBook are all independent of any particular programming language. The problems in this eBook can be solved with **any programming language**: assembler, C, C++, Delphi, Perl, Python, Eiffel, Pascal, Java, Forth, etc. In some places a particular programming language may be mentioned, such as Java, but this is just in discussion. Many of the problems can even be solved using **SQL** and the results stored in database tables.

Latest Version

The latest version of this document will be on my web site. The current location is:

http://moosesoftware.netau.net/

if my web site link (above) no longer works (my web site has moved hosts, etc), then my latest web site will be posted here in a Reddit forum dedicated to my software:

https://www.reddit.com/r/MoosesSoftware/

License

Solutions

Solutions are not made available to any of these questions. It's all about learning and exploring and researching.

If you cannot solve a particular problem, discuss the problem with others and work together (see **Introduction**).

If there is enough demand, I might give additional hints and tips and prepare partially worked solutions – enough to explore everything and map everything out so that you can finish it off from there.

Acknowledgements

I would especially like to acknowledge this book and author:

- ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

which contained some really interesting problems to solve, on which some of the questions below are based.

Contacting the Author

If you would like to suggest additional problems for this eBook, or if you have any issues, errors, corrections, feedback, suggestions, etc, please visit my web page:

http://moosesoftware.netau.net/

and click the "**Contact Me**" link in the menu (on the right, at the top of page), and my email address is there. ☺

Programming Practice Questions

Visualising our data – a simple Bar Chart tool

Charts can be a great way to see results, because they make trends much more obvious and aid understanding.

Your task is to create a bar graph method that will be useful in many of the questions below.

Start off simple:

```
barGraph (20);
```

would simply draw 20 asterisks:

```
********************
```

Hints:

- It is highly recommended that you generate a string of the output bar chart and then print this string to the screen – don't print the asterisks individually to the screen - because this will be much more useful / adaptable if you need to display the bar chart in a GUI application, a dialog, etc later on.

It's a start, but we can do better. Most of the time, you will want to plot many values, so modify your method to accept an array of doubles:

```
double myData []  = {3, 30, 8, 27.5, 19, 75};

barGraph (myData);
```

would produce:

```
***
******************************
********
***************************
*******************
***************************************************************************
```

Which is great, but what if you need to plot large values – you don't want lines full of asterisks if you need to plot the value 1,000. Alter your barGraph method to scan through the data before plotting it and determine the maximum values and scale the values we need to plot accordingly.

```
double myData []  = {300, 30, 8, 27.5, 219, 75};
```

the largest value is 300, and say we want the maximum to be 80 asterisks, so we scale all other values by: 80.0 / 300.0 and truncate the decimals and draw that many asterisks for each value:

```
300 * 80.0 / 300.0      = 80    → so we draw 80 asterisks
30 * 80.0 / 300.0       = 8     → so we draw 8 asterisks
8 * 80.0 / 300.0        = 2     → so we draw 2 asterisks
and so on …
```

OK, so far so good ?

Another change we need to make is to be able to plot positive and negative values. To do this, we need to scan through our data, determine the minimum and maximum values, calculate the origin, and shift and scale the values we need to plot accordingly.

For example:

double myData [] = {-200, -120, -8, -27.5, 199, 300};

Min: -200
Max: 300

So, the range of values we are plotting is 200 + 300 = 500. If we want our asterisk line to be 80 chars max, we need to divide all values by 500 for plotting. Also, to handle positive and negative numbers, we need to move / shift the values to the right so that the origin, value 0, is about 200 / 300 = 2/3 of the way across the screen.

For each value to be plotted, you need to calculate the spaces you need to indent, and then asterisks you need to display. Don't Panic – this might sound complex, but the calculations are quite straight forward.

OK, let's work it through with the 1st value: **-200**

numSpaces	= Abs ((-200 – Min) / (Max – Min)) * 80	= 0.0
numAsterisks	= Abs (-200 / (Max – Min)) * 80	= 32.0

so we indent by 0 spaces and then draw 32 asterisks and then a vertical bar | to represent the origin.

Our origin is 32 spaces across the screen. Remember that !

OK, onto the next value: **-120**

numSpaces	= Abs ((-120 – Min) / (Max – Min)) * 80	= 12.8
numAsterisks	= Abs (-120 / (Max – Min)) * 80	= 19.2

so we indent by 12 spaces and then draw 19 asterisks and then a vertical bar | to represent the origin.

OK, let's move to a positive number so you can see how it works there: **199**

numSpaces	= 32 (recall our Origin value from above)	
numAsterisks	= Abs (199 / (Max – Min)) * 80	= 31.84

so we indent by 32 spaces and then a vertical bar | to represent the origin, and then and then draw 31 asterisks.

See the pattern ? It's pretty easy once you see the pattern. ☺

One last change that is useful is to have labels for each bar, so you can see the real values being plotted. To do this, you could use String.format to output a suitably formatted value prior to each bar line.

Here's the finished bar graph with added labels (modified to use 65 instead of 80 max width to fit here):

```
-200.00 ***************************|
-120.00            ****************|
  -8.00                           *|
 -27.50                         ***|
 199.00                            |***************************
 300.00                            |*****************************************
```

Congratulations ! ☺ If you have completed this task, you now have a general purpose bar charting tool that will help you visualise data and be very useful in the future.

Further improvement suggestions:

- Make a BarGraph class so that this functionality is really easy to re-use in any project – you are not copying and pasting the barGraph method(s) into your other classes – creating horrible duplication of code – instead you are re-using the BarGraph class whenever you need it.
- Add any other public or private methods that may be useful … or that may help break down the complex barGraph method into separate tasks, each in their own method …. Maybe these:
    - getBarLine (value) ?
    - getSpacing (value) ?
    - getAsterisks (value) ?
- Anyway, will leave this to your imagination / needs.

Euler's number - e

Euler's number – **e** – is very important in mathematics, for example see this video:

e (Euler's Number) – Numberphile, https://www.youtube.com/watch?v=AuA2EAgAegE

e can be calculated as follows:

e = 1 / 0! + 1 / 1! + 1 / 2! + 1 / 3! + 1 / 4! + ...

! indicates factorial. 0! = 1. 1! = 1. 2! = 1 x 2 = 2. 3! = 1 x 2 x 3 = 6. 4! = 1 x 2 x 3 x 4 = 24. And so on.

Create a method which accepts an integer and calculates and returns the factorial of that integer. e.g. fact(4) would return 24.

Calculate **e** to 5 decimal places.

How many factorial terms do you need ? How fast is this converging ? Why ? Explore, try it out and see, have fun, learn – that's what these questions are all about !

Change your program over to use Java's **BigDecimal** class. Note everything changes ! Search on the internet and find out how to use Java's **BigDecimal**, and then convert your code to use this class.

Calculate e to 1,000 decimal places. Make sure you answer agrees with **Euler's Number - e - to 1,000 Decimal Places** on page 39.

Display the 526^{th} decimal place of e. What's the easy way to do this ?

Hints / additional information:

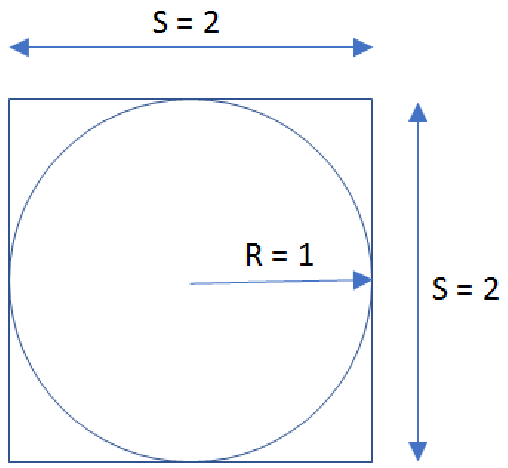
- See **Correct to N Decimal Places ? How can you tell ?** on page 39.
- See **Euler's Number - e - to 1,000 Decimal Places** on page 39.

Pi Calculator – Dart Board method

Pi is another very important constant in mathematics. It crops up everywhere, even when there are no circles in sight. I was very interested in Pi when I was younger, and wrote my first Pi Calculator on my **Apple][** way back in 1980. Here's how I did it …

Note: this is a horrible method for calculating Pi – very slow to converge – but it is easy and fun to program ! ☺

Imagine I have a dart board setup with a square around it – so the dartboard (the circle) fits perfectly inside the square:



Area of Square = S * S = 2 * 2 = 4;

Area of Circle = Pi * R * R = Pi x 1 x 1 = Pi

If I blind fold myself and throw darts, and keep track of:

- Ns = number of darts that land inside the square (or circle).
- Nc = the number of darts that land inside the circle.

then the ration of Nc / Ns should be the same as the ratio of the circle and square's areas: Pi / 4

That is:

Pi = 4 * Nc / Ns

Write a program that throws virtual darts at this dartboard plus square target, and keep track of the Nc and Ns and use these values to calculate Pi to 5 decimal places.

More information:

- For Java, use Math.random() - which generates really poor quality random numbers in the range 0.0 to 0.999999...
- Generate a random X coordinate between 0.0 and 1.0.
- Generate a random Y coordinate between 0.0 and 1.0.
- Calculate the distance from the origin:
- D = SQRT (X * X + Y * Y)
- Is D <= 1.0, then the dart lies on or inside the circle, so add 1 to Nc.
- Add 1 to Ns.
- Calculate the **Pi = 4.0 * Nc / Ns**
- Display to the screen. Maybe only display the value of Pi every 100 or 1,000 or more iterations ?
- Keep going until Pi is correct to 5 decimal places. See **Correct to N Decimal Places ? How can you tell ?** on page 39.

```
*** Sample Output:

Iteration Pi (estimate)
1. 4.0
```

```
2. 4.0
3. 4.0
4. 4.0
5. 3.2
. 3.3333333333333335
7. 3.4285714285714284
8. 3.5
9. 3.5555555555555554
10. 3.2
11. 3.272727272727273
12. 3.0
13. 2.769230769230769
14. 2.857142857142857
15. 2.933333333333333
16. 3.0
17. 3.0588235294117645
18. 2.888888888888889
19. 2.9473684210526314
20. 3.0
21. 2.857142857142857
22. 2.909090909090909
23. 2.9565217391304346
24. 3.0
25. 3.04
::: etc
```

Note:

- If you need a more exact value of Pi, use Math.PI in Java.

How many iterations are required to get Pi correct to 5 decimal places ? See **Correct to N Decimal Places ? How can you tell ?** on page 39.

Questions / tasks:

- Change over to using better random numbers, e.g. Java's SecureRandom class:

```
import java.security.SecureRandom;
SecureRandom generator = new SecureRandom();
double val = generate.nextDouble();
```

- How many iterations are required to get Pi correct using these much better random numbers to 5 decimal places ?
- Which random numbers do you have the most faith ? Why ?
- Would you recommend this method to calculate Pi to 1,000 decimal places ? Millions decimal places ? Why ?
- Find 2 other methods that are better for calculating Pi to millions decimal places. Why are they better ?

Google's Recruitment Billboard (2004) - Wanna Work for Google ??? (Expired)

Hints and tips:

- Calculate **e** to 1,000 decimal places (for starters, you might need more ???).
- Store the digits of e in an array.
- Work through the digits of e and check whether each digit is a prime number, and keep a count of the number of prime digits in a row.
- When you find 10 prime digits in a row, you've found the answer. Huzzah !! ☺

Notes:

- HINT: answer ends with …. 391
- The solution web site no longer exists. The problem was solved very quickly - it is an extremely easy problem in the scheme of things.
- I was very disappointed when I heard about this problem, I was hoping for a gruelling scan through trillions of digits of **e**, so I wrote some code to do this and ran it expecting it to take a while, and the answer popped out straight away. Boo Google !!! BOO !!! ☺

Cable Car – Position and Velocity

Based on a great question from one of Mike O's old university books:

- Chapter 4, Exercise 5, p88-125, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

A 1000 meter cable is stretched between two towers, with a supporting tower (a 3rd tower) midway between the two end towers. The velocity of the cable car depends on its position on the cable. When the cable car is within 30 meter of a tower, its velocity (in meter/sec) is given by:

V = 2.425 + 0.00175 * D ^ 2

Otherwise (if the cable car is NOT within 30 meter of a tower), its velocity (in meter/sec) is given by:

V = 0.625 + 0.12 * D - 0.00025 * D ^ 2

where:

- D is the distance to the nearest tower.

Print a table starting with the cable car starting at the first tower and moving to the last tower in increments of 10 meter. At each location along the cable, print the nearest tower number (1 = first, 2 = middle, 3 = last), the distance from the first tower, and the velocity of the cable car.

To ensure you understand the problem, why not enter the velocity formulas and some test data into an Excel spreadsheet. Make sure you are getting values that match mind in the sample table below.

Cable Car Report				
Distance from Start	Nearest Tower Number	Distance to Nearest Tower	Cable Car Velocity	Cable Car Velocity Graph
(m)		(m)	(m/sec)	
0	1	0	2.43	****
10	1	10	2.6	****
20	1	20	3.13	*****
30	1	30	4	*******
40	1	40	5.02	*********
50	1	50	6	***********
60	1	60	6.92	************
70	1	70	7.8	**************
80	1	80	8.63	****************
90	1	90	9.4	*****************
100	1	100	10.13	******************
110	1	110	10.8	********************
120	1	120	11.42	*********************
130	1	130	12	**********************
140	1	140	12.53	***********************
150	1	150	13	************************
160	1	160	13.42	*************************
170	1	170	13.8	*************************

180	1	180	14.12	**************************
190	1	190	14.4	***************************
200	1	200	14.63	***************************
210	1	210	14.8	***************************
220	1	220	14.92	***************************
230	1	230	15	****************************
240	1	240	15.02	****************************
250	1	250	15	****************************
260	2	240	15.02	****************************
270	2	230	15	****************************
280	2	220	14.92	***************************
290	2	210	14.8	***************************
300	2	200	14.63	***************************
310	2	190	14.4	***************************
320	2	180	14.12	**************************
330	2	170	13.8	*************************
340	2	160	13.42	*************************
350	2	150	13	************************
360	2	140	12.53	***********************
370	2	130	12	**********************
380	2	120	11.42	*********************
390	2	110	10.8	********************
400	2	100	10.13	******************
410	2	90	9.4	*****************
420	2	80	8.63	****************
430	2	70	7.8	**************
440	2	60	6.92	************
450	2	50	6	***********
460	2	40	5.02	*********
470	2	30	4	*******
480	2	20	3.13	*****
490	2	10	2.6	****
500	2	0	2.43	****
510	2	10	2.6	****
520	2	20	3.13	*****
530	2	30	4	*******
540	2	40	5.02	*********
550	2	50	6	***********
::: etc				

Mike O’s bonus marks / additions:

- When you have fully answered the base question above ...
- Add a bar graph of velocity to the table. Use your BarGraph class to plot the data. See **Visualising our data – a simple Bar Chart** on page 4.
- Make the towers, and the distances between them, items in array(s) or other suitable data structure(s). Make it so that adding or removing towers is as easy as changing the elements in the data structure and recompiling.

- **Use an Objects approach:** make a **CableCar** data class with id, position (along cable), etc as class fields and getVelocity(), etc as class methods. Have different types of cable car – with a different velocity calculation for each (alter the above formula). e.g. a "Service Car" that is bigger and slower, a 3 person chair (which is light and fast).
- Expand your program to have multiple cable cars of various types, and the all cars slow down when any of them is within "slow down" range of a tower.
- Write the results, bar graph, etc out to a text file.
- Make a GUI version of the program, with one or more sliders to indicate the cable car position. Add functionality to enable the user to Add, Edit, Delete, Move towers, add / delete cable cars, etc
- Explore, learn, have fun !

Radioactive Decay of Thorium

Based on a great question from one of Mike O's old university books:

- Chapter 2, 2.6 Application - Radioactive Decay, p34, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.
- With Thorium reactors looking increasingly likely on humanity's horizon, this question is very relevant !

The radioactive decay of thorium is given by:

N = No * e ^ (-0.693 * t / 1.650E16)

where:

- **No** is the initial amount of thorium,
- **e** is Euler's number = 2.7182818284590452353603 (to 25 decimal places),
- **t** represents the time elapsed (in seconds).

When t = 0, N = No and no decay has occurred. As t increases, the amount of thorium remaining decreases.

How far it the future will half the thorium remain ?

Hint: it's more than a million years !

- To ensure you understand the problem, why not enter the velocity formulas and some test data into an Excel spreadsheet. Make sure you are getting values that match mind in the sample table below.
- You probably want to output data every 100,000 years or so ....

Years	Pct Thorium Remaining
100,000	0.999868
200,000	0.999735
300,000	0.999603
400,000	0.99947
500,000	0.999338
600,000	0.999206
700,000	0.999073
800,000	0.998941
900,000	0.998809
1,000,000	0.998676
1,100,000	0.998544
::: etc	

Mike O's bonus marks / additions:

- When you have fully answered the base question above ...
- Add a bar graph of velocity to the table. Use your BarGraph class to plot the data. See **Visualising our data – a simple Bar Chart** on page 4.
- Do research on other elements: Plutonium, Uranium, etc.

- Adapt the formula above to calculate their half lives. (Simple changes needed).
- **Use an Objects approach:** make an **Element** class with element name, half life, etc.
- Expand your program to allow the user to select the element, and display results, bar graph, etc for that element.
- Write the results, bar graph, etc out to a text file.
- Make a GUI version of the program.
- Explore, learn, have fun !

Gravity: Planets, Stars, and Black Holes

The force of gravity between 2 objects can be calculated as follows:

F = G x M1 x M2 / R^2

where:

- F = force of gravity
- G = Newton's Gravitational Constant = 6.674×10^{-8} m^3 / Kg / s^2
- M1 = mass of object in Kg
- M2 = mass of object in Kg
- R = distance between the objects in meters (m).

You need to use the internet to research the masses, distances, etc. Shock horror – having to do your own research ???? Yes, just like in the real world. ☺ You are likely to find conflicting values, just go with average distances, etc. We are not concerned with figures that are accurate to 10 decimal places here – just ball park figures are fine.

Calculate the gravity between the Sun and Earth at these distances:

- Earth's normal distance from the Sun
- If Earth was moved to the distance of Mars from the Sun
- If the Earth was moved to the distance of Neptune from the Sun

If the Sun became a black hole (all of the mass of the Sun shrunk down to a point in space), what would be the gravity between the Black Hole Sun and Earth at these distances:

- Earth's normal distance from the Sun
- If Earth was moved to the distance of Mars from the Sun
- If the Earth was moved to the distance of Neptune from the Sun

What do you conclude from this ?

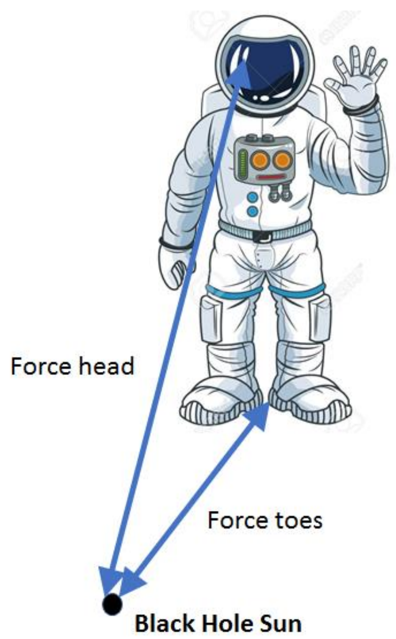
If the Sun became a black hole tomorrow, what effect would this have on the orbit of Earth ?

If you were in a space suit and suspended 30m, 10m and 1m above the Black Hole Sun, so that your feet pointed "down" and your head was "up", what would be the force of gravity:

- At your feet
- At your head

What would happen to your body ? What do astronomers call this effect ?

Calculate and bar graph the increase of gravity as you proceed towards the Sun from the orbit of Pluto. Select a suitable display increment so the results fit on a computer screen.



Mike O's bonus marks / additions:

- When you have fully answered the base question above ...

- Create a bar graph of gravity as you move between 2 objects – from a suitable distance, until they are very close. What shape is the graph ? What can you say about the gravity changing with respect to distance ?
- **Use an Objects approach:** make an **AstronomicalObject** data class with name, radius, mass, ... and suitable getters, setters, constructors, etc, and a gravity () method that calculates the gravity between 2 **AstronomicalObject**s at a specific distance apart.
- Write the results, bar graph, etc out to a text file.
- Make a GUI version of the program. Work out how to display images of the planets, etc and display an image for each object.
- Explore, learn, have fun !

Gravity: Escape Velocity and Event Horizons

WIP

Mike O’s bonus marks / additions:

- Write the results, bar graph, etc out to a text file.
- Make a GUI version of the program.
- Explore, learn, have fun !

Rocket Ship – Take Off and Landing

Based on a great question from one of Mike O's old university books:

- Question 26, Chapter 4, pp130-131, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.
- With space travel being the key to humanity's long term survival, this question is very relevant !

A small test rocket is being designed for use in testing a retrorocket that is intended to permit "soft landings" (similar to what SpaceX did ~35+ years after this question was written). The designers have derived the following equations that they believe will predict the performance of the test rocket:

acceleration = 4.25 - (0.015 * t ^ 2.0) + (6.07 * t ^ 2.751) / 9995

velocity = 4.25 * t - ((0.015 * t ^ 3.0) / 3.0) + (6.07 * t ^ 3.751) / (3.751 * 9995)

height = launchPadHeight + (4.25 * t ^ 2.0) / 2.0) - ((0.015 * t ^ 4.0) / 12.0) +

(6.07 * t ^ 4.751) / (4.751 * 3.751 * 9995)

where:

- **t** represents the time elapsed (in seconds).
- The launch pad is 90 meters above ground.

Assume all units are in metric (m, m/sec^2, m/sec).

Write a program that "test flies" the rocket, calculating and displaying the height, acceleration, and velocity every 2 seconds throughout the flight until either 100 seconds has passed, or the Rocket is at or below ground level. i.e. generate the table below.

Questions / tasks:

- To ensure you understand the problem, why not enter the velocity formulas and some test data into an Excel spreadsheet. Make sure you are getting values that match mind in the sample table below.
- After how much time does the rocket land ?
- What is the rocket's maximum speed ?
- What is the rocket's maximum acceleration ?
- Generate a bar graph of the Rocket's speed – see below.

Time (sec)	Acceleration (m/sec^2)	Velocity (m/sec)	Height (m)
0	0	0	90
2	4.19	8.46	98.48
4	4.04	16.71	123.7
6	3.79	24.55	165.05
8	3.48	31.84	221.55
10	3.09	38.41	291.92
12	2.66	44.17	374.65
14	2.17	49	467.98
16	1.66	52.84	570

18	1.11	55.62	678.63
20	0.55	57.29	791.72
22	-0.01	57.83	907.02
::: etc			

Sample Bar Chart of Rocket's Speed:

```
                                           |****
                                           |*********
                                           |**************
                                           |******************
                                           |**********************
                                           |**************************
                                           |****************************
                                           |*******************************
                                           |********************************
                                           |*********************************
                                           |**********************************
                                           |*********************************
                                           |********************************
                                           |*******************************
                                           |****************************
                                           |*************************
                                           |**********************
                                           |******************
                                           |*************
                                           |*********
                                           |***
                                          *|
                                     ******|
                               ************|
                          *****************|
                    ***********************|
               ****************************|
           ********************************|
       ************************************|
   ****************************************|
 ******************************************|
*******************************************|
*******************************************|
 ******************************************|
    ***************************************|
        ***********************************|
               ****************************|
                       ********************|
                                  *********|
                                           |
```

Mike O's bonus marks / additions:

- Explore, learn, have fun !

Space Race: 3 Spaceships to Alpha Centauri

Based on:

- Mike O's own imagination. I enjoy thinking about stuff like this.

Multiple spaceships are sent out from Earth: destination Alpha Centauri our nearest stellar neighbour, 4.2 light years away.

Space Craft #1: Impulse Engine

- Immediate acceleration - no manoeuvring around sun necessary.
- Starting velocity = velocity in orbit around the Earth = 25.0 Km/sec
- Acceleration: 0.01 G
- Mass: 170,000 Kg

Space Craft #2: Solar Sail

- needs 2 years to manoeuvrer into the right orbit around the sun and extend the sails.
- Starting velocity = velocity in tight orbit around the Sun = 265.0 Km/sec
- Acceleration: 60 G until 1 billion Km from the sun and then 0 from then on (simplified to how this would really work).
- Mass: 50,000 Kg

Space Craft #3: Solar Sail + Impulse

- heavier and slower because of additional engine / hardware.
- needs 2 years to manoeuvrer into the right orbit around the sun and extend the sails.
- Starting velocity = velocity in tight orbit around the Sun = 195.0 Km/sec
- (Yes, slower velocity because of extra mass)
- Acceleration: 16 G until 1 billion Km from the sun and then 0 from then on (simplified to how this would really work).
- (Yes, slower acceleration because of extra mass)
- Mass: 200,000 Kg
- At that point, jettison the solar sail arms / hardware (reducing mass by 30,000 Kg), and Impulse engine can be fired up
- Acceleration: 0.01 G (300 million Km from Sun and beyond)

Assume all distances are linear, and all velocities are in the correct direction (no additional manoeuvring required).

Some useful formulas:

- S = UT + 1/2 AT^2
- V = SQRT (U^2 + 2 AS)
- F = MA (see *** below)

where:

- T: time in seconds
- U: initial velocity in Km/Sec
- A: Acceleration in Km/Sec/Sec
- S: Distance travelled so far in Km.
- F: Force
- M: Mass

Do whatever research you need to fill in the missing details / pieces of the puzzle.
Yep, just like in real life ! :)

*** Is mass of the space ship relevant to this question ? You have been told the acceleration (in G) ? What does this mean ? Do we need to worry about F = MA in this question ?

```
*** Sample Output:

Space Ship #1:
---------------
 0 Years                  0 Km  0.00%      25.000 Km/sec  0.00% Light
 1 Years     49,569,758,590 Km  0.01%   3,118.687 Km/sec  0.10% Light
 2 Years    196,702,231,359 Km  0.05%   6,212.375 Km/sec  0.21% Light
 3 Years    441,397,418,309 Km  0.11%   9,306.062 Km/sec  0.31% Light
 4 Years    783,655,319,438 Km  0.20%  12,399.750 Km/sec  0.41% Light
 5 Years  1,223,475,934,747 Km  0.31%  15,493.437 Km/sec  0.52% Light
:::
etc
```

Questions / Tasks:

- Where is your start point ? Earth ? The Sun ? Does it really matter ?
- Plot the distance / velocity for each spaceship. Select suitable increments. e.g. display spaceship status information every year.
- Which space ship arrives first at Alpha Centauri ?
- At Alpha Centauri, what speed are the spaceships going ?

Hints and Tips:

- Start with Ship #1
- Explore the calculations in Excel and your programming language - just for the first 100 seconds, and then for the first 100 days. Ensure you are getting the same values.
- Be very careful of your units of measure - ensure your conversions are correct.
- e.g. If acceleration is 5.4 G, then this is 5.4 x the force of gravity on Earth at sea level = 5.4 x 9.81 m/sec/sec. You might need to convert this to Km/sec/sec for your space travel calculations.
- Do the full calculation for Ship #1.
- **HINT: Ship #1 should arrive at Alpha Centauri after approx 90 years.**
- Make sure you get an answer close to this before moving onto the other ships.

Mike O's bonus marks / additions:

- Explore, learn, have fun !
- Can you make your code better, easier to re-use ?
- Create a **SpaceShip class** with suitable class fields (acceleration, starting velocity, etc) and suitable methods (getVelocityAtSecs (), getDistanceAtSecs (), etc).
- Be careful not to let Ship #2 and #3 complicate things. Maybe you can split these into multiple "pseudo" ships.
    - e.g. For Ship #2: it only accelerates for the first part of the journey, and then it coasts the rest of the way. Create Ship 2A for the first part of the journey, and then Ship 2B when it coasts through the bulk of the journey to the destination. For Ship #3: create Ship 3A for the first part of the journey, and then when the solar sails detach, create another ship - Ship 3B using Ship 3A's final velocity, etc as starting values.
- Make sure your space ship velocity cannot exceed 95% of the speed of light (because very strange things happen after this speed, and we don't want to complicate things).
- What if the solar sail space ship acceleration decreased with square of the distance from the Sun ?

Visualising Data - Interference Pattern

Based on a great question from one of Mike O's old university books:

- Example 8.3, p258-263, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

In this application, we are interested in studying the interference pattern produced by two sinusoidal wave generators operating in phase.

Examples:

- two speakers driven by the same amplifier.
- two radio antennas powered by the same transmitter.

We have a function that defines a displacement Z that describes the interference of the sources. If we are dealing with surface waves, Z would be the actual displacement of the surface, which could be water. If we are talking about sound waves, Z could refer to pressure.

R1 = distance between current point X,Y and the 1st source of interference.
= Sqrt ((X + source1X) ^2 + (Y + source1Y) ^2)
R2 = distance between current point X,Y and the 2nd source of interference.
= Sqrt ((X + source1X) ^2 + (Y + source1Y) ^2)
Z = displacement (the interference) at the current point.
= cos (R1) + cos (R2)

Plot the interference pattern between -10 and +10 for X and Y where the two wave sources are located at -1,-1 and +1,+1. We want to represent the displacement Z with characters such that the character becomes darker as the displacement increases.

Sample output:

Interference Pattern: 40w x 25h
```
,++++++++++,,,,,,,,,,,,,,,,,,++%@#####@%
++++++++,,,,,,,,,,,,,,,,,,,,,,,,+%@####@
++++++,,,,,,,,,,,,,,,,,,,,,,,,,,,,+%@###
+++++,,,,,,,,,,,,+++++++,,,,,,,,,,,,+%##
+++,,,,,,,,,,++++%%%%%@@%%%+,,,,,,,,,,%@
++,,,,,,,,+++++%%%%%@@@####@%+,,,,,,,,,%
+,,,,,,,,++++++++++%%%%@@####@%+,,,,,,,,
,,,,,,,++++++++++,,,,,,+%%@####@%,,,,,,,
,,,,,,,+++++++,,,,,,,,,,,,+%####@%,,,,,,
,,,,,,+++++++,,,,,,,,,,,,,,,%@###@+,,,,,
,,,,,++%%%++,,,,,,,,,,,,,,,,,%@##@%,,,,,
,,,,,+%%%%++,,,,,,,,,,,,,,,,,+%@@@%+,,,,
,,,,,+%%%%+,,,,,,,++++,,,,,,,,%%@@%+,,,,
,,,,,+%@@%%,,,,,,,,++++,,,,,,,+%%%%+,,,,
,,,,,+%@@@%+,,,,,,,,,,,,,,,,,++%%%%+,,,,
,,,,,,%@##@%,,,,,,,,,,,,,,,,,++%%%++,,,,
,,,,,,+@###@%,,,,,,,,,,,,,,,+++++++,,,,,
,,,,,,,%@####%+,,,,,,,,,,,,+++++++,,,,,,
+,,,,,,,%@####@%%+,,,,,,++++++++++,,,,,,
%,,,,,,,,+%@####@@%%%%++++++++++,,,,,,,,
@%,,,,,,,,,+%@####@@@%%%%%+++++,,,,,,,,+
#@%,,,,,,,,,,+%%%@@%%%%%++++,,,,,,,,,,++
```

```
###%+,,,,,,,,,,,,+++++++,,,,,,,,,,,,++++
####@%+,,,,,,,,,,,,,,,,,,,,,,,,,,,,+++++
%@####@%+,,,,,,,,,,,,,,,,,,,,,,,,+++++++
```

Mike O’s bonus marks / additions:

- Explore, learn, have fun !

Factorials - Stirling's Approximation

Based on a great question from one of Mike O's old university books:

- Chapter 3, Q19, p136, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

Factorials can be estimated using Stirling's Approximation:

N! = Sqrt (2 * PI * N) * (N / e)^N

Where:

- PI = 3.14159265358979
- e = 2.718281828459045

Write a program to calculate the factorials for all integers between 1 and 20. This program should calculate:

- the exact Factorial values using iteration or recursion,
- the approximate Factorial values using Stirling's Approximation,

and display the results in a table. For each factorial, calculate and display the error%:

Error% = (Exact Value - Approximate Value) / Exact Value * 100.0

What happens to the error% as N increases ?

Benchmark the calculation of factorials. Calculate the above factorials 1000 times and time each method. Which is fastest ? By how much ?

Sample Output:

```
 N             Factorial               Stirling Factorial          Error%
----  --------------------------  -----------------------------  -------------------
 1                            1                          0.922     7.7862991104211%
 2                            2                          1.919     4.0497824255509%
 3                            6                          5.836     2.7298401442356%
 4                           24                         23.506     2.0576036129446%
 5                          120                        118.019     1.6506933686750%
 6                          720                        710.078     1.3780299108077%
 7                        5,040                      4,980.396     1.1826223886417%
 8                       40,320                     39,902.395     1.0357255638475%
 9                      362,880                    359,536.873     0.9212762230081%
10                    3,628,800                  3,598,695.619     0.8295960443939%

::: etc
```

Stirling's method is brilliant and gives increasingly accurate answers as N increases. (Error -> 0 as N -> Infinity).

There are lots of other brilliant Factorial methods, so give even better answers than Stirling's method.

References / further reading:

- https://en.wikipedia.org/wiki/Stirling%27s_approximation

- http://www.luschny.de/math/factorial/approx/SimpleCases.html

Mike O’s bonus marks / additions:

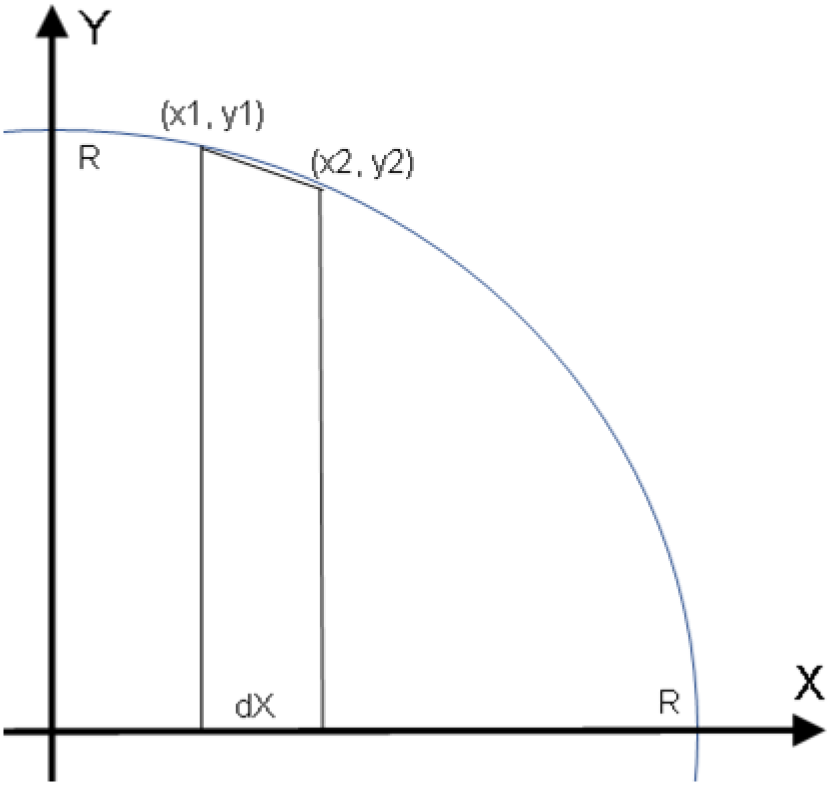
- Explore, learn, have fun !

Pi Calculator – Trapezoids and Circles – ****   Work In Progress

Based on a great question from one of Mike O's old university books:

- Chapter 9, Q26 , p325., ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

Write a program to compute PI using double precision (or better) variables. The algorithm to be used should compute the area of a quarter circle with a radius of 1.0 and multiply that area by 4.0 to get an approximation of PI. To calculate the area of a quarter circle, you need to sum the areas of subsections (slices) of the quarter circle. The area of a subsection is approximately equal to the area of a trapezoid.



The diagram above shows a single slice.

- i = current slice number (1, 2, 3, 4, ...)
- DX = R / NumSlices
- Xi = R - i x DX
- Y1 = (R^2 - Xi^2)^0.5
- Y2 = (R^2 - (Xi + DX)^2)^0.5

Area of a subsection (slice) = SUB = (Y2 + Y1) / 2.0 * DX

Area Quarter Circle = AREAQ = Sum SUB for each slice

Area circle = PI = AreaQ x 4.0

(because Radius = 1.0)

Area Circle = PI * R^2 = PI   (because R = 1.0)

For this question, use 2000 slices.

Compare your calculated value of PI against the real value (PI = 3.14159265358979) and calculate and display the error%:

errorPct = (PIactual - PIcalc) / PIactual * 100.0

Vary the number of slices from 1000.0 up to 10,000.0 in steps of 1,000 and display the value of PI calculated with each number of slices and the error% in the calculated PI value.

```
Slices         PI              Error%
-------  ------------------  ---------
 1,000  3.1415554669110293  0.001184%
 2,000  3.1415795059119660  0.000419%
 3,000  3.1415854968639048  0.000228%

 ::: etc
```

Vary the number of slices from 1.0 and double the number of slices each iteration and keep looping while the number of slices is less than 1 million. Display the value of PI calculated with each number of slices and the error% in the calculated PI value.

```
Slices          PI                    Error%
---------  ------------------  --------------------
        1  2.0000000000000000  36.3380227632418000%
        2  2.7320508075688770  13.0361218394416410%
        4  2.9957090681024403  4.6436187492561630%
        8  3.0898191443571736  1.6480019831169570%
       16  3.1232530378277410  0.5837681005872281%

::: etc
```

Mike O’s bonus marks / additions:

- Explore, learn, have fun !

Trend Line by Method of Least Squares

Based on a great question from one of Mike O's old university books:

- Chapter 6, Exercises 6.5 and 6.6, pp182-187, Structured Fortran 77 for Engineers and Scientists, D. M. Etter. (C) 1983. ISBN: 0-8053-2520-4

I have rewritten the question and discussion below to make it a lot clearer and more focused, and suitable for use in tutorial questions or perhaps even an Assignment for students.

The following data represents the load-deflection of a coil spring, where:

- Load is the weight applied to the spring in Kg, and
- Length is the length of the spring in cm after the Load has been applied to the spring.

Load	**Length**
0.28	**6.62**
0.50	**5.93**
0.67	**4.46**
0.93	**4.25**
1.15	**3.30**
1.38	**3.15**
1.60	**2.43**
1.98	**1.46**

Example: for a load 0.28 Kg, the length of the spring is 6.62 cm.

As the load increases, the length of the spring decreases (because the spring is compressed and forced down).

Store the above data in a suitable array and calculate and display the equation of the "line of best fit" (aka "trend line"). The "**Method of Least Squares**" is a standard technique for determining the equation of a straight line from a set of data, also called the "line of best fit" and "trend line".

Recall that the equation of a straight line is:

Y = M X + C

where M is the slope (or gradient) of the line and C is the Y intercept (where the line crosses the Y axis).

The slope and Y intercept for the "line of best fit" (aka "trend line") can be calculated from data using the following equations:

sumX = X1 + X2 + ... + Xn
sumY = Y1 + Y2 + ... + Yn
sumXY = X1 * Y1 + X2 * Y2 + ... + Xn * Yn
sumXX = X1 * X1 + X2 * X2 + ... + Xn * Xn
slope = (sumX * sumY - n * sumXY) / (sumX * sumX - n * sumXX)
Y intercept = (sumY - slope * sumX) / n

(n is the number of data points)

If the data points and trend line were plotted on a piece of graph paper, or using computer software like a spreadsheet, it would be possible to see how closely the line fitted the data. Another way to see how well the line fits the data is the take an X value and plug it into the "trend line" equation, thus calculating a Y value. This new Y value, designated Y', would be an estimate of the value of Y for a given X value.

For example, suppose the least squares technique yielded the following trend line equation:

Y' = 4.2 X - 3.1

For a given data point (1.0, 0.9) (i.e. X = 1.0, Y = 0.9) the estimated (calculated) Y value is:

Y' = 4.2 * 1.0 - 3.1 = 1.1

Display all points in the array, and for each point calculate and display the estimated (calculated) Y value, Y', using the trend line equation.

The residual for a data point is the difference between the actual value of Y and the estimated value of Y:

residual = Y - Y'

The residual for the data point above is 0.9 - 1.1 = -0.2.

The sum the squares of the residuals for all data points is called the Residual Sum, and this provides an estimate of the quality of fit of the trend line to the actual data, without needing to plot the actual data. When the trend line is a perfect fit for the data (the data is perfectly linear), then the Residual Sum will be zero. As Residual Sum increases, the less the data can be estimated by the trend line. Calculate and display the Residual for all data points and the Residual Sum.

Now that we have out trend line equation and a measure of how good the line is, we can use the trend line to make estimates and fill in gaps.

Use your trend line to answer the following questions:

- The natural length of the spring (i.e. the length when there is no load).
- The load when Spring is 5 cm long

```
Load    Length  Length    Length
(X)      (Y)     (Y)       (Y)
Actual  Actual  Estimate  Residual
------  ------  --------  --------
0.28    6.62     6.24      0.38
0.50    5.93     5.59      0.34
0.67    4.46     5.10     -0.64
0.93    4.25     4.33     -0.08
1.15    3.30     3.69     -0.39
1.38    3.15     3.02      0.13
1.60    2.43     2.37      0.06
1.98    1.46     1.26      0.20

Residual Sum (Error) =  0.88
```

Estimates based on Trend Line:

- Length of Spring (no load) = 7.06
- Length of Spring with 2.1 Kg load = 0.91
- Load when Spring is 5 cm long = 0.70

Mike O’s bonus marks / additions:

- Explore, learn, have fun !

Graph of Current in Circuit

Based on a great question from one of Mike O's old university books:

- Question 50, Chapter 8, p284, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

Prepare a plot of the current A(t) of the following circuit:

```
----------------------------------------
|                                      |
|                                 /    |
|                               /      |
-/\/\/\/\----oOOo-----:|:|----/    -----
Resistor  Inductor  Charge  Switch
```

where:

A(t) = AMAX x e^(-250 x t) x cos (7061.4 x t)

e = 2.718281828, AMAX = 0.07176, and a time t increment of 1.69E-05.

Start with t = 0 and plot 200 points of the function.

Sample output:

```
                                    |*****************************************
                                    |****************************************
                                    |***************************************
                                    |**************************************
                                    |***********************************
                                    |*********************************
                                    |******************************
                                    |**************************
                                    |**********************
                                    |******************
                                    |**************
                                    |**********
                                    |*****
                                    |
                                 ***|
                            ********|
                        ************|
                    ****************|
                ********************|
            ************************|
         ***************************|
      ******************************|
    ********************************|
  **********************************|
 ***********************************|
************************************|
************************************|
************************************|
 ***********************************|
  **********************************|
    ********************************|
      ******************************|
        ****************************|
           *************************|
               *********************|
                  ******************|
                      **************|
```

```
                   **********|
                       ******|
                           **|
                             |**
                             |******
                             |**********
                             |*************
                             |*****************
                             |********************
                             |***********************
                             |**************************
                             |****************************
                             |******************************
                             |*******************************
                             |********************************
                             |********************************
                             |********************************
                             |********************************
                             |*******************************
                             |*****************************
                             |****************************
                             |*************************
                             |***********************
                             |********************
                             |*****************
                             |*************
                             |**********
                             |******
                             |***
                             |
                         ****|
                      *******|
                  ***********|
               **************|
            *****************|
         ********************|
       **********************|
     ************************|
   **************************|
  ***************************|
 ****************************|
*****************************|
*****************************|
*****************************|
 ****************************|
  ***************************|
    *************************|
      ***********************|
        *********************|
          *******************|
             ****************|
                *************|
                   **********|
                      *******|
                          ***|
                             |
                             |**
                             |******
                             |*********
                             |************
                             |**************
                             |*****************
                             |*******************
                             |*********************
                             |***********************
                             |************************
                             |*************************
                             |**************************
                             |**************************
                             |**************************
                             |*************************
                             |************************
                             |***********************
                             |*********************
```

```
                       |********************
                       |*****************
                       |***************
                       |************
                       |**********
                       |*******
                       |****
                       |*
                      *|
                   ****|
                *******|
             **********|
           ************|
         **************|
      *****************|
     ******************|
   ********************|
  *********************|
 **********************|
***********************|
***********************|
***********************|
***********************|
 **********************|
  *********************|
   ********************|
     ******************|
       ****************|
         **************|
           ************|
              *********|
                *******|
                   ****|
                     **|
                       |
                       |***
                       |*****
                       |********
                       |**********
                       |************
                       |**************
                       |****************
                       |*****************
                       |*******************
                       |********************
                       |********************
                       |*********************
                       |*********************
                       |********************
                       |********************
                       |*******************
                       |******************
                       |*****************
                       |***************
                       |*************
                       |***********
                       |*********
                       |*******
                       |*****
                       |**
                       |
                     **|
                   ****|
                 ******|
               ********|
             **********|
           ************|
         **************|
        ***************|
       ****************|
      *****************|
     ******************|
     ******************|
     ******************|
```

```
******************|
******************|
 *****************|
  ****************|
   ***************|
    **************|
      ************|
       ***********|
         *********|
           *******|
             *****|
               ***|
                  |
                  |*
                  |***
```

Mike O’s bonus marks / additions:

- Explore, learn, have fun !

Isothermal Metal Plate - Temperature Distribution in Metal Plate

Based on a great question from one of Mike O's old university books:

- Example 9.2, Chapter 9, p297-304, ***Structured Fortran 77 for Engineers and Scientists***, D. M. Etter, (C) 1983. ISBN: 0-8053-2520-4.

In this application we consider the temperature distribution in a thin metal plate as it reaches a point of thermal equilibrium. The plate is constructed so that each edge is "isothermal", or maintained at a constant temperature. The temperature of an interior point on the plate is a function of the temperature of the surrounding material. If we consider the plate to be similar to a grid, then a two-dimensional array could be used to store the temperatures of the corresponding points on the plate.

The isothermal temperature at the top, bottom, left, and right [edges of the metal plate] would be given. The interior points are initially set to some arbitrary temperature, usually zero. The new temperature of each interior point is calculated as the average of its four surrounding [adjacent] points, as follows:

```
   T1
T4 T0 T2      T0 = (T1 + T2 + T3 + T4) / 4
   T3
```

After computing the new temperatures for each interior point, the difference between the old temperatures and the new temperatures is computed. If the magnitude of a temperature change is greater than some specified tolerance value, the plate is not yet in thermal equilibrium, and the entire process is repeated.

Since we will use only one array for the temperatures, as we change one temperature, this new value will affect the change in adjacent temperatures. The final results will also be slightly different depending on whether the changes are made across the rows or down the columns.

Use a tolerance value of 0.2.

Mike's Notes:

- A nice little problem. I LOVED solving problems like this when I was at school and uni, and decades years later, I still LOVE solving these problems.
- I don't think it is possible for a metal plate (or a plate made with any other material) to have the suggested properties. i.e., keeping the edges at a constant temperature - does not change - must be impossible unless some special heating / cooling machine is maintaining the edge temperatures.

Sample Output:

```
Initial Temperatures:
100.0   100.0   100.0   100.0
100.0     0.0     0.0   200.0
100.0     0.0     0.0   200.0
200.0   200.0   200.0   200.0

Temperature Iteration #1:
100.0   100.0   100.0   100.0
```

```
100.0    50.0    87.5   200.0
100.0    87.5   143.8   200.0
200.0   200.0   200.0   200.0

Temperature Iteration #2:
100.0   100.0   100.0   100.0
100.0    93.8   134.4   200.0
100.0   134.4   167.2   200.0
200.0   200.0   200.0   200.0

::: etc
```

Mike O’s bonus marks / additions:

- When you have fully answered the base question above ...
- **Use an Objects approach:** make a **IsothermalPlate** data class with grid length, grid width, a 2D array to hold the temperature of each grid cell, ... and suitable getters, setters, isStable(), etc. Add a range of constructors to initialise the plate to different starting scenarios – all edge grid cells at a given temperature, random temperatures, etc. Add a blowTorch() and chill () methods that apply heat or cold to particular grid cells.
- Make a GUI version of the program and draw the grid as a range of buttons (or whatever) – display the temperature value and colour code each button (black = cold, dark red = warm, light red = hot, yellow/white = very hot, etc. Add start, stop, pause, reset buttons. When start is clicked, use a timer to update the grid cells and stop when the stability has been reached.
- Explore, learn, have fun !

Cryptocurrency – “Bitcoin”, Blockchain, Mining - **** Work In Progress

WIP

Mike O’s bonus marks / additions:

- Explore, learn, have fun !

Appendix A – FAQ

Correct to N Decimal Places ? How can you tell ?

How are you going to even know when the value you are calculating is correct to N decimal places ?

If you know the exact value you need to calculate or compare against, for example you are calculating Pi correct to 100 decimal places and you have accepted as correct Pi to 100 decimal places or more, then you can compare directly against this and see how close you are like this:

if (Math.abs (myValue – exactValue) < 0.00001)

// the value is correct to 4 decimal places.

Make sure you take the ***absolute value*** of the difference between the 2 values – this will check if you are very close to the exact value - either just under or just over.

If you don't know what the exact value is, then you need to be very careful, otherwise you might accept a value that isn't correct. In this case, all we can do for now is watch the data being generated. For example, if our calculates / approximations are generating numbers like this:

- **1.343**4787112
- **1.343**5166334
- **1.343**4988121
- :::: etc

Then only values after the 3rd decimal place are changing, and we can very cautiously suggest that the value might be correct to 3 decimal places,

Euler's Number - e - to 1,000 Decimal Places

2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852
516642742746639193200305992181741359662904357290033429526059563073813232862794349076323382988
075319525101901157383418793070215408914993488416750924476146066808226480016847741185374234544
243710753907774499206955170276183860626133138458300075204493382656029760673711320070932870912
744374704723069697720931014169283681902551510865746377211125238978442505695369677078544996996
794686445490598793163688923009879312773617821542499922957635148220826989519366803318252886939
849646510582093923982948879332036250944311730123819706841614039701983767932068328237646480429
531180232878250981945581530175671736133206981125099618188159304169035159888851934580727386673
858942287922849989208680582574927961048419844436346324496848756023362482704197862320900216099
023530436994184914631409343173814364054625315209618369088870701676839642437814059271456354906
13031072085103837505101157477041718986106873969655212671546889570350116