File failed to load: https://cs455-usc.updog.co/res/styles/js/MathJax/jax/output/HTML-CSS/fonts/TeX/fontdata.js

CS/EE 455 Final Exam

6/21/18, 1-3 pm for in-class students, alternative as-scheduled for DEN students

Please note the following:

\* submit your work on D2L as usual, under the 'Exam' slot; there is nothing to write/submit 'on paper'

\* you are allowed to use your own laptop, and can use ANY software you want - editors, IDEs, shells, compilers, viewers etc.

\* you are allowed to look up ANYTHING ANYWHERE - online, your own files, books, cheat sheets.. anything at all!

\* you can use any device/medium (laptop, tablet, smartphone, flash drive, SD card..)

\* you can't get anyone else's help with any part of any question; a LOT of trust is being placed on you to do the right thing, please honor it! It is NOT worth getting ahead using unethical means.

\* please observe the time limit of 2 hours

\* my cell # is 310-490-8095, and my email is saty@usc.edu - proctors are welcome to call/email during the exam if they have additional questions or concerns

\* at the start of the test, you might possibly get an email from me, with an updated version of the exam (with more details, clarification, etc), so be on the lookout for this

There are four [two C++, two Java] questions below. Each is worth 5 points.

**Have fun!** Hopefully it is a straightforward exam, and hope you enjoy thinking about and coding the solutions.

Q1. Here is a classic trig identity:

https://cs455-usc.updog.co/m18_CsML00Agi/hw/Fn-0x4eb0/pics/ident.jpg

**Write C++ code (call it identity.cpp) that 'proves' the formula**, for angles 0,1,2,3....359 degrees. main() needs to call two functions - LHS() and RHS(), which will calculate the LHS and RHS (left and right hand sides) of the identity when supplied an angle:

double LHS(double angle) {

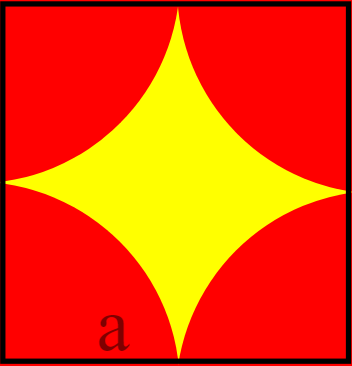
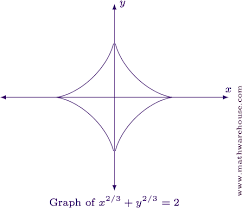
}// LHS()

double RHS(double angle) {

}// RHS()

You would conclude that LHS and RHS are identical if they differ by less than 1e-10 (0.0000000001), in which case your program needs to print TRUE (otherwise it will print FALSE). You need to use the ? ("question mark") operator in your code.

Q2. The yellow shape below, which is constructed using quarter circles centered at each corner of a square whose side is 'a', happens to be a good approximation to the 'astroid' curve shown underneath it:

If a=2, what is the area (numerical value) of the yellow shape? **Write a C++ program (calcArea.cpp) that uses Monte-Carlo ('dart throwing') to compute and output the answer** [you CANNOT use any explicit area formulae (eg. πr2) to arrive at the result!]. Be sure to throw at least 10,000 'darts'. And, note that you wouldn't need to use the astroid formula shown above in any way - the astroid shape is provided only as a loose visual comparison.

Q3. **Write a Java class called sineTable**, which will store sin() values in increments of 10 degrees, between 0 and 360 (ie. 0,10,20,30...360 degrees), and will return a linearly interpolated value for a user-requested angle. Eg. if the user asks for sin(45), it will use sin(40) and sin(50) [both of which it has already calculated and stored], to calculate an interpolated value for sin(45) (in this case, the result would be the mean of sin(40) and sin(50). In cheap electronics, such a technique might be used, where angles are stored in ROM, and interpolated on the fly (instead of storing closer-spaced angles' sin() values, or computing sin() from scratch).

FYI: given a lower value l, upper value u and an interpolation fraction f (where f is a fraction in 0..1), the (linearly) interpolated result is

interp = l + f\*(u-l)

Here is what your sineTable class needs to have:

\* sineValues: an ArrayList list [for storing sin(0), sin(10)..]

\* sineTable(): constructor - this will fill the sineValues array mentioned above

\* public String toString() - this will print the sineValues array (eg. for debugging), like so:

0: 0.0

10: 0.17364818458302766

20: 0.3420201565241952

30: 0.5000000182457416

40: 0.6427876312056138

50: 0.7660444656897871

60: 0.8660254248528056

...

...

300: -0.8660252984425806

310: -0.7660443031799403

320: -0.6427874375339252

330: -0.4999997992968278

340: -0.34201991895070877

350: -0.1736479356035202

360: 2.5282041413652845E-7

\* calcValue(): given an angle in degrees, will carry out the linear interpolation and return the result. Eg. given 45 (degrees) as input, it needs to return 0.7044160484477004 [note that actual sin(45) is 0.7071, but ours is just an interpolated approximation that uses sin(40) and sin(50) to calculate its answer].

To get the value for 45 degrees as shown above, the user of the sineTable class would simply write (please do this in your psvm so you can verify your code works!):

public static void main(String[] args) {

sineTable s = new sineTable();

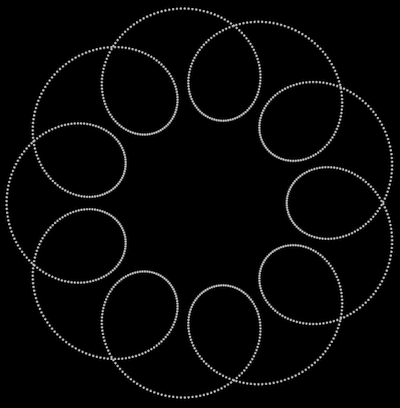
//System.out.println(s); // for debugging

System.out.println(s.calcValue(45)); // should print 0.7044160484477004

}

NOTE/HINT: Math.sin() expects angles in radians, not degrees!

Q4. Using Java, **write a curveGenerator class**, which will output a curve.wrl (VRML) file that contains a set of small, adjacent spheres (of radius 0.1) that help create this pretty, symmetric curve:



Here is the curve's (parametric) equation in pseudocode form:

R=9., r=1., a=4.

for t from 0.00 to pi\*18 in steps of 0.05

x = (R+r)\*cos((r/R)\*t) - a\*cos((1+r/R)\*t)

y = (R+r)\*sin((r/R)\*t) - a\*sin((1+r/R)\*t)

z = 0.0

Load your curve.wrl into ['instantplayer'](http://www.instantreality.org/downloads/), admire the result, generate and submit (in addition to curveGenerator.java) a screenshot in .png or .jpg format.

Submission checklist:

\* Q1. identity.cpp

\* Q2. calcArea.cpp

\* Q3. sineTable.java

\* Q4. curveGenerator.java, screenshot