COM S 227
Fall 2019
Miniassignment 2: Recursion
40 points

Due Date: Friday, November 22, 11:59 pm (midnight)
5% bonus for submitting 1 day early (by 11:59 pm November 21)
10% penalty for submitting 1 day late (by 11:59 pm November 23)[1]
No submissions accepted after November 23, 11:59 pm

General Information

This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus, http://www.cs.iastate.edu/ cs227/syllabus.html#ad , for details.
You will not be able to submit your work unless you have completed the *Academic Dishonesty policy acknowledgement* on the Homework page on Canvas. Please do this right away.
If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Note: This is a miniassignment and the grading is automated. If you do not submit it correctly, you will receive at most half credit.

Overview

In this assignment you will implement a few recursive methods. Most of these methods can be implemented iteratively, some of them more easily or more efficiently than a recursive solution; *you will not receive credit for these problems if you submit an iterative solution!* The purpose of this assignment is to give you practice writing recursive methods. Similarly, some of these methods are "solved" or mostly-solved by Java library methods (in particular `toInt()` and `Integer.parseInt()`, and `pow()` and `Math.pow()`); *you will not receive credit for these problems if you submit a solution that simply calls an equivalent or very similar Java library method!*

Advice

Before you write any code for a method, think about the bottom of the recursion (the "base case") and how the problem shrinks with each recursive call. Most of these methods are simply an `if` statement (the base case) and a `return` statement with a recursive call—`isMatched()` is somewhat more involved—so once you figure these two things out, your most of the way to a solution

My code's not working!!

Developing recursive methods can be hard, especially when you're used to thinking iteratively. If you are getting errors, a good idea is to take a simple, concrete example, and trace execution of your code by hand (as illustrated in section 6.2 of the text) to see if the code is doing what you want it to do. You might draw a call tree to see what your stack looks like as your program executes. You can also trace what's happening in

[1] It's break; don't turn it in late! If necessary, work extra hard this week, so you don't have to work while you're traveling, trying to relax, catching up with old friends, etc.

the code by temporarily inserting `println` statements to check whether variables are getting updated in the way you expect. (Remember to remove the extra `println`'s when you're done!)

Overall, the best way to trace through code with the debugger, as we practiced in Lab 6. Learn to use the debugger effectively, and it will be a lifelong friend.

Always remember: one of the wonderful things about programming is that within the world of your own code, you have absolute, godlike power. If the code isn't doing what you want it to do, you can decide what you really want, and make it so. **You are in complete control!**

(If you are not sure what you want the code to do, well, that's a different problem. Go back to the "Advice" section.)

Testing and the SpecChecker

A SpecChecker will posted shortly that will perform an assortment of functional tests. As long as you submit the assignment correctly, your score will be exactly the score reported by the specchecker.

However, when you are debugging, it is much more helpful if you have a simpler test case of your own that reproduces the error you are seeing.

Since no test code is being turned in, you are welcome to post your tests on Piazza for others to use and comment on.

Documentation and style

Since this is a miniassignment, the grading is automated and in most cases we will not be reading your code. Therefore, there are no specific documentation and style requirements. However, writing a brief descriptive comment for each method will help you clarify what it is you are trying to do.

If you have questions

For questions, please see the Piazza Q & A pages and click on the folder `miniassignment2`. If you dont find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag `miniassignment2`. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Java examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have Java code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it. Be sure you have stated a specific question; vague requests of the form "read all my code and tell me whats wrong with it" will generally be ignored.

Of course, the instructors and TAs are always available to help you. See the Office Hours section of the syllabus to find a time that is convenient for you. We do our best to answer every question carefully, short of actually writing your code for you, but it would be unfair for the staff to fully review your assignment in detail before it is turned in.

Any posts from the instructors on Piazza that are labeled "Official Clarification" are considered to be part of the spec, and you may lose points if you ignore them. Such posts will always be placed in the Announcements section of the course page in addition to the Q&A page. (We promise that no official clarifications will be posted within 24 hours of the due date.)

What to turn in

Note: You will need to complete the "Academic Dishonesty policy questionnaire," found on the Homework page on Blackboard, before the submission link will be visible to you.

Please submit, on Canvas, the zip file that is created by the SpecChecker. The file will be named `SUBMIT_THIS_mini2.zip`. and it will be located in the directory you selected when you ran the SpecChecker. It should contain one directory, `mini2`, which in turn contains one file, `Mini2.java`.

Always LOOK in the zip file the file to check what you have submitted!

Submit the zip file to Canvas using the Miniassignment2 submission link and verify that your submission was successful. If you are not sure how to do this, see the document "Assignment Submission HOWTO" which can be found in the Piazza pinned messages under Syllabus, office hours, useful links.

We strongly recommend that you just submit the zip file created by the specchecker. If you mess something up and we have to run your code manually, you will receive at most half the points.

> We strongly recommend that you submit the zip file as created by the specchecker. If necessary for some reason, you can create a zip file yourself. The zip file must contain the directory `mini2`, which in turn should contain the file `Mini2.java`. You can accomplish this by zipping up the `src` directory of your project. The file must be a zip file, so be sure you are using the Windows or Mac zip utility, and not a third-party installation of WinRAR, 7-zip, or Winzip