CSE 250 PA0: Hello, World of Data Structures! **Fall 2019**

Due: Friday, 9/13/2019 before 5:00PM

Total points = 80

1 Objectives

In this assignment you will:

- Read and understand the course AI policy and late policy.
- Review (or learn) Scala basics.
  - Where to find/how to use Scala docs.
  - How to edit/compile/debug code in Scala.
  - How to find/read the Scala documentation and use standard libraries.
  - How to read and parse a file in Scala.
  - How to write a file in Scala.
- Write basic tests using ScalaTest.

2 Useful Resources

Review the lecture notes and provided example code for some insight into the Scala syntax. You will also want to read the Scala references provided below:

- ScalaTest basics (FlatSpec): http://www.scalatest.org/user_guide/writing_your_first_test
- scala.collections.mutable.Map:
  https://www.scala-lang.org/api/current/scala/collection/mutable/Map.html
- Scala File I/O (Scala Cookbook Excerpt):
  https://buffalo.box.com/v/cse250-ScalaCookbookIO
- Scala API: https://www.scala-lang.org/api/current/index.html
- Scala Resources: https://docs.scala-lang.org/
- Scala Tour: https://docs.scala-lang.org/tour/tour-of-scala.html
- Scala Collections: https://docs.scala-lang.org/overviews/collections-2.13/introduction.html
- Scala Exercises: https://www.scala-exercises.org/
- Scala Map:
  https://docs.scala-lang.org/overviews/collections-2.13/maps.html

3 Late Policy

The policy for late submissions on assignments is as follows:

- More than 5 days before the deadline: +5 points + 100% of what you earn on your best submission.
- Up to 5 days before the deadline: +1 point per day + 100% of what you earn on your best submission.
- On the deadline: 100% of what you earn on your best submission.
- One day late: 25 point deduction (lower total points by 25).
- Two days late: 50 point deduction (lower total points by 50).
- $>$ 2 days late: 0 points.

A day is considered as a 24 hour period counting from the deadline time. From this policy, you will see that all assignments must be submitted within two days past the assigned deadline. The date of submission is that of your latest submission made, even if this is not your highest scoring submission. For assignments with multiple component submissions, only 1 penalty will be assessed based on the file submitted last. If a staggered deadline is given (e.g., two components due one week apart), the earlier deadline will be a hard deadline and no late submissions will be accepted for the first component.

You will have the ability to use three grace days throughout the semester, and at most two per assignment (since assignments are not be accepted beyond two days late). Using a grace day will negate the 25 point penalty for one day of late submission, but will not allow you to submit more than two days late. Please plan accordingly. You will not be able to recover a grace day if you decide to work late and your score was not sufficiently higher. **Grace days are automatically applied** to the first instances of late submissions, **and are non-refundable**. For example, if an assignment is due on a Friday and you make a submission on Saturday, you will automatically use a grace day, regardless of whether you perform better or not. **Be sure to test your code before submitting**, especially with late submissions, **in order to avoid wasting grace days**.

Keep track of the time if you are working up until the deadline. Submissions become late after the set deadline. Keep in mind that **submissions will close 48 hours after the original deadline** and you will no longer be able to submit your code after that time.

4 AI Policy Overview

As a gentle reminder, please re-read the academic integrity policy of the course. I will continue to remind you throughout the semester and hope to avoid any incidents.

4.1 What constitutes a violation of academic integrity?

These bullets should be obvious things not to do (but commonly occur):

- Turning in your friend's code/write-up (obvious).
- Turning in solutions you found on Google with all the variable names changed (should be obvious). This is a copyright violation, in addition to an AI violation.
- Turning in solutions you found on Google with all the variable names changed and 2 lines added (should be obvious). This is also a copyright violation.

- Paying someone to do your work. You may as well not submit the work since you will fail the exams and the course.
- Posting to forums asking someone to solve the problem.

  Note: Aggregating every [stack overflow answer|result from google|other source] because you "understand it" will likely result in full credit on assignments (if you aren't caught) and then failure on every exam. Exams don't test if you know how to use Google, but rather test your understanding (i.e., can you understand the problems to arrive at a solution on your own). Also, other students are likely doing the same thing and then you will be wondering why another person that you don't know has your solution.

You should know that seeking solutions to the assignment does not fall under solving the problem yourself. Things that may not be as obvious:

- Working with a tutor who solves the assignment with you. If you have a tutor, please contact me so that I may discuss with them what help is allowed.
- Sending your code to a friend to help them. **If another student uses/submits your code, you are also liable and will receive the same punishment**.
- Joining a chatroom for the course where someone posts their code once they finish, with the honor code that everyone needs to change it in order to use it.
- Reading your friend's code the night before it is due because you just need one more line to get everything working. It will most likely influence you directly or subconsciously to solve the problem identically, and your friend will also end up in trouble.

The assignments should be solved individually with only assistance from course staff and allowed resources. You may discuss and help one another with technical issues, such as how to get your compiler running, etc. It is not acceptable that you both worked together and have nearly identical code. If that is going to be a problem for you, don't solve the problems in that close of proximity.

4.2 What collaboration is allowed?

There is a gray area when it comes to discussing the problems with your peers and **I do encourage you to work with one another to solve problems**. That is the best way to learn and overcome obstacles. At the same time you need to be sure you do not overstep and not plagiarize. Talking out how you eventually reached the solution from a high level is okay:

> I used a stack to store the data and then looked for the value to return.

but explaining every step in detail/pseudocode is not okay:

> I copied the file tutorial into my code at the start of the function, then created a stack and pushed all of the data onto the stack, and finished by popping the elements until the value is found and use a return statement.

The first example is OK but the second is basically a summary of your code and is not acceptable, and remember that you shouldn't be showing any code at all for how to do any of it. Regardless of where you are working, you must always follow this rule: **Never come away from discussions with your peers with any written work, either typed or photographed, and especially do not share or allow viewing of your written code.**

If you have concerns that you may have overstepped or worked closely with someone, please address me prior to deadlines for the assignment. We will address options at that point.

4.3 What resources are allowed?

With all of this said, please feel free to use any [files|examples|tutorials] that we provide directly in your code (with proper attribution). Feel free to directly use anything from lecture or recitations. You will never be penalized for doing so, but must always provide attribution/citation for where you retrieved code from. Just remember, if you are citing an algorithm that is not provided by us, then you are probably overstepping.

More explicitly, you may use any of the following resources (with proper citation/attribution in your code):

- Any example files posted on the course webpage (from lecture or recitation).
- Any code that the instructor provides.
- Any code that the TAs provide.
- Any code from the tour of Scala (https://docs.scala-lang.org/tour/tour-of-scala.html)
- Any code from Scala Collections (https://docs.scala-lang.org/overviews/collections-2.13/introduction.html)
- Any code from Scala API (https://www.scala-lang.org/api/2.13.0/)
- Additional references may be provided as the semester progresses, but only those provided publicly by course staff are allowed to be utilized. These will be listed on Piazza under Resources.

Omitting citation/attribution will result in an AI violation (and lawsuits later in life at your job). This is true even if you are using resources provided.

Lastly, **if you think you are going to violate/have violated this policy, please come talk to me ASAP so we can figure out how to get you on track to succeed in the course**. This policy on assignments is here so that you learn the material and how to think yourself. There is no benefit to submitting solutions (which likely exist in some form).

5 Collaboration Policy

The policy for collaboration on assignments is as follows:

- All work for this course must be original individual work.
- You must follow the limits on collaboration as defined in the AI policy (i.e., no work written while together/shared code/etc.).
- You must identify any collaborators (first and last name) on every assignment. This can be in a comment at the top of your code submissions or on the first page at the top of your written work beside your name.

All references must be cited using a comment containing a direct link to the resource as well as a brief description of what was used. For example, if you reference the textbook, a page number and description is sufficient. If you copy example code from the Scala language API, then include the link to the class page within the API as well as where the example code resides.

6 Instructions

Unit Testing

Answer the following questions by submitting your answers to Autolab via the "PA0 Unit Testing" assignment. **Expect this section to take around 2-3 hours of reading through documentation and thinking of what to test.**

For each of the following problems, you will create a ScalaTest class to test each of the specified functions. Your code will be tested against various implementations of the function, which may be correct or incorrect. When given a correct solution, your tests should all pass. When given an incorrect solution, at least one of your tests should fail. You should only test the function on the given range of inputs. Each of the functions is defined within an `object` in the package `cse250.pa0.objects` named `Functions`. Each test class should be in the package `cse250.pa0.tests` or testing will not perform correctly.

Problem 1. (10 points) Complete the definition of the class `GenNumTests` that tests the function `genNum` defined by:

```
def genNum(n: Int): Int = { ??? }
```

where:

- The range of n is $[0, \ldots, 1000]$.
- Every number returned should be positive.

Problem 2. (10 points) Complete the definition of the class `GenSeqTests` that tests the function `genSeq` defined by:

```
def genSeq(n: Int): Seq[Int] = { ??? }
```

where:

- The range of n is $[0, \ldots, 1000]$.
- Every item in the returned sequence should be even.
- The size of the returned sequence should be n.

Problem 3. (10 points) Complete the definition of the class `FunThreeTests` that tests the function `funThree` defined by:

```
def funThree(n: Int): Int = { ??? }
```

where:

- The range of n is $[0, \ldots, 1000]$.
- `FunThree` should be a non-decreasing function.

Note: a function f is non-decreasing if $\forall i, f(i) \leq f(i+1)$.

Problem 4. (10 points) Complete the definition of the class `CompSumTests` that tests the function `compSum` defined by:

```
def compSum(n: Int): Long = { ??? }
```

where:

- The range of n is $[1, \ldots, 50000]$.
- The value returned should be $\sum_{i=1}^{n} i$.

Property Dataset

Answer the following questions by submitting your answers to Autolab via the "PA0 Property Tax" assignment. Submit your solution in a single `.scala` file.

Expect this section to take 5-10 hours of setting up your environment, reading through documentation, and planning, coding, and testing your solution.

Problem 5. (40 points) We will be making use of the 2017-2018 tax assessment dataset (https://data.buffalony.gov/Government/2017-2018-Assessment-Roll/bxmp-ux8w) from the Buffalo OpenData portal (https://data.buffalony.gov/) this semester. You can download a copy of the dataset by clicking the "Export" button followed by "CSV." Your task with this dataset will be to cleanup the data file. There are a number of unnecessary columns that are not of interest to us, so we will create an output file with these columns removed. We also want to cleanup the data by removing entries that are missing certain information.

You should complete the definitions for the following functionality all within the same `object` called `TaxEntryProcessor`. You will use the `TaxEntry` class to store the data for each record. Note that all info is stored within a `Map` object. The `Keys` should be the header string for the column and all values should be strings from the cells.

5(a) **(20 points)** Define the Scala function

```
sanitizeData(filename: String): Unit
```

that performs the following tasks:

(1) Opens the data file at the path given in the input string `filename`.

(2) Fixes the data as follows:

- Remove the necessary columns from each row (given in the table below).
- Omit rows that are missing a zip code. (These correspond to utilities and the likes)

(3) Writes the modified rows to a new output file with the following properties: (*) a properly formatted CSV file, (*) maintains the ordering of the original headers (less the removed headers), and (*) the same name as the input with `"-updated"` appended to the end.

- For example:
  - If the string contained in `filename` is `"tax-data"`, the output filename should be `"tax-data-updated"`.
  - If the string contained in `filename` is `"data.csv"`, the output filename should be `"data.csv-updated"`.

The file format is a CSV (comma separated values). CSV files are a way to represent columns of data by separating entries within a row by a comma. Each line represents a separate row of cells. Two special cases arise that you must handle:

- If a cell contains a comma (`,`) within, the cell contents are enclosed in double quotes (`"`) at the start and end. For example: `Comma, Cell` would be stored as `"Comma, Cell"`.
- If a cell contains a double quote (`"`) within, the cell contents are enclosed in double quotes (`"`) at the start and end AND each double quote (`"`) is duplicated. For example: `The "Best" Around` would be stored as `"The ""Best"" Around"`.

It is also possible that a cell contains both.

The following columns should be removed:

Column Number	Heading
1	SBL
2	TAX DISTRICT
8	PREVIOUS PROPERTY CLASS
9	OWNER1
10	OWNER2
11	MAIL1
12	MAIL2
13	MAIL3
14	MAIL4
21	ZIP CODE (4-DIGIT)
22	DEED BOOK
23	DEED PAGE
25	ROLE SECTION
32	BUILDING STYLE
33	HEAT TYPE
34	BASEMENT TYPE
40	CENSUS TRACT
41	CENSUS BLOCK GROUP
42	CENSUS BLOCK

When you are finished, there should be 27 columns remaining in the dataset.

5(b) **(10 points)** Define the function

```
computeMostExpensiveEntry(filename: String): TaxEntry
```

that (1) Opens the data file at the path given in the input string `filename`, and (2) Returns the `TaxEntry` for the most expensive location assessed in the database. You should assume that the data file is the dataset with the appropriate columns and rows removed (assume you have the correct output from part (a)).

5(c) **(10 points)** Define the function

```
computeOldestEntry(filename: String): TaxEntry
```

that (1) Opens the data file at the path given in the input string `filename`, and (2) Returns the entry for the oldest property assessed in the database. You should assume that the data file is the dataset with the appropriate columns and rows removed (assume you have the correct output from part (a)).

Note: you should review the dataset/your code results to ensure your answer makes sense.

To represent a single data record, you may use the structure `cse250.assignments.classes.TaxEntry` provided in the code skeleton:

```
class TaxEntry {
  val infoMap: Map[String, String] = new HashMap[String,String]

  override def toString: String = infoMap.toString()
```

PA0/src/cse250/assignments/objects/TaxEntry.scala

Note that any changes to this file containing `TaxEntry` will be overwritten when your code is graded, so any changes you make within will be reverted.

Suggested Approach

1. Download the PA0 skeleton files (PA0-handout.zip) onto your computer and unzip them. Make sure to remember where you unzipped them to.
2. Import the PA0 project files into IntelliJ and configure environment:
   (a) Open IntelliJ.
   (b) Select `Open` (possibly `File > Open`).
   (c) Navigate to the folder PA0 that you extracted from the handout zip. Click OK.
3. Download the 2017-2018 tax data set and move it to the `PA0/data` folder.
   - The data file can then be opened via the filename: `data/2017-2018_Assessment_Roll.csv`
   - I recommend you make a smaller test file of entries to work on. To do this, make a copy of the assessment role file and then remove all of the lines after the first 10 or 100, etc., then save the file.
   - **It is not recommended to open the file in Excel as there may be unintended formatting side-effects upon saving.**
   - When viewing the `.csv` file in IntelliJ, I recommend **not** installing the plugin so that you can continue viewing it as text instead of the view that would be provided by software like Excel. This is beneficial so you can see how the data you are manipulating looks.
4. Update the copyright statement with your name and UBIT name in the necessary files.
5. Begin working on the problems requested.
   - Note that when working on reading the csv data files, the last column of the data contains a comma and should be treated as a single entry. There may be other data entries that contain commas, as well, so be sure you think about how to handle this (look at the data set in a text editor/IntelliJ to see what the format is).
   - To check that this works, try pausing the program using the debugger/breakpoints to check that the data is being read in as you expected.
   - All code can be tested via the `Main` method in `cse250.pa0.objects` and in the various test files. To run the `Main` method, right-click the `Main` file and select `Run Main`. To run the tests, right-click the folder `test` and select `Run 'ScalaTests in 'test'...'`. You are welcome to add more testing functions at your discretion, both in the `Main` object as well as in `TaxEntryProcessorTests`.

Allowed header/library usage

- You may not use other libraries that are not already imported in the given code.

If you find there is a library that is omitted here that you would like to use, please ask on Piazza for permission to use it. Otherwise, all other includes are not allowed.

Submission

For each section (PA0 Unit Tests and PA0 Property Tax), you will be allowed 5 submissions to each, without penalty. Starting from the 6th submission, you will receive

- a 5 points per submission deduction from your score on the respective assignment.

Note: your score is what you receive on your latest submission. If you receive a score and then resubmit, even if you receive a lower score/0 points that will be your score for the assignment.

Also note: if you submit early/late, the bonus/penalty would be awarded against the entirety of the assignment, not just the single part you submitted early/late.

Creating Your Submission

Your submission files should all be zipped together and submitted as one. We will extract the relevant files and grade them based on the problem you submit to. To produce a zip file:

1. Right-click PA0 in IntelliJ and select `Show in Explorer`.
2. Zip the entire contents into one file.
    - For Windows: Right-click PA0 > Send to > Compressed (zipped) folder.
    - For OsX: Right-click PA0 > Compress "..." (... should likely be PA0 , but I don't have a Mac to confirm).
    - For Linux: Right-click PA0 > Compress ... > Select .zip extension. (may differ based on your distro).
    - You can also, from IntelliJ, select Open in Terminal and then use the zip command-line tool (untested, use at own risk).
3. Upload your zip file to the appropriate submission you wish to be graded on.

7 Change Log

- 9/5: Fixed typo in package for Unit Test objects. Was `cse250.pa0.handout` but should be `cse250.pa0.objects` which is reflected in the handout code.