# CS106 W21 - Assignment 03

Due: Friday, February 5, 11:59 PM

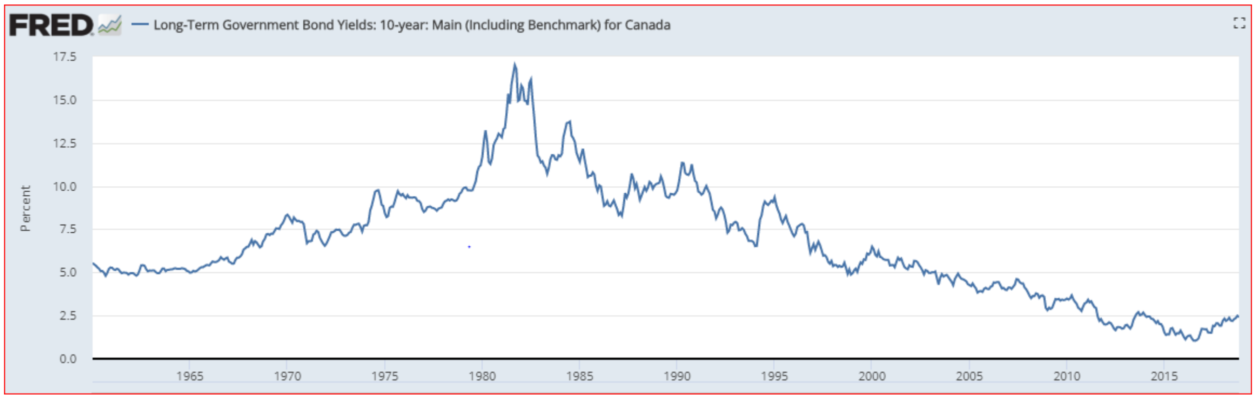
Assignment 3 is graded out of 31 marks.

---

## QUESTION ONE:

**Question One: Canadian Historical Interest Rates: Requirements and Grading [ 16 marks]**

The website https://fred.stlouisfed.org contains various data. It has data for the monthly interest rates in Canada from 1960 to 2018. Shown in a graph the data looks as follows:



The graph shows, for example, that interest rates in the very early 1980's were historically very high.

We have the data in a txt file named interestRates.txt. The data file interestRates.txt has four columns representing the month, day, year and interest rate. The data file interestRates.txt has 707 lines and starts as follows:

```
1  1  1960     5.54
2  1  1960     5.46
3  1  1960     5.39
4  1  1960     5.29
5  1  1960     5.21
6  1  1960     5.07
…
…
…
```

Write a sketch that reads interestRates.txt and draws a graph that is similar to the graph above.

There starter code is at: https://openprocessing.org/sketch/1080169. The text file "interestRates.txt" is included in the starter code. You may also use any code from the course such as the sea ice examples shown in the video lectures:

January sea ice example: https://openprocessing.org/sketch/1077561

Monthly sea ice example: https://openprocessing.org/sketch/1077652

You can make the assumption that the interest rate is always between zero and 20%.

1) [ 4 marks] Download a photo from your camera and name that "background". The photo should be of a Canadian coin or Canadian $20 bill or something that represents money. It doesn't matter exactly the photo is, but it should represent currency, or somehow represent Canadian money. Display this photo somewhere on the canvas. Perhaps it will be a background for the whole canvas, or perhaps it will be small and in the upper left corner.

   Put a title on the graph.

2) [ 4 marks] The line on the graph must be drawn correctly. You must use beginShape(), vertex(), and endShape() to draw the graph. It doesn't have to look like the image above exactly, but it must be correct.

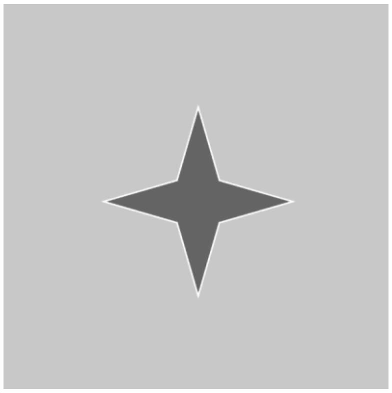
3) [ 4 marks] You must have a margin on the bottom (x-axis) to show the years in increments of 5. You must use map() to do this. It doesn't have to look like the image above exactly, but it must be correct.

4) [ 4 marks] You must have a margin on the left (y-axis) to show the interest rate in increments of 2.5%. You must use map() to do this. It doesn't have to look like the image above exactly, but it must be correct.

QUESTION TWO:

**Question Two: Stars Requirements and Grading [ 10 marks]**

You are to mimic the behavior you see in this video: https://vault.cs.uwaterloo.ca/s/iiETmyZQwnYmi6j



There starter code is the Regular Polygons code that we saw in the video lectures on slides 46-47: https://openprocessing.org/sketch/1079214

The only code you need to modify is the code between beginShape() and endShape(). No other code needs to be modified.

1) [ 4 marks] Modify the code so that there are "inner" vertices as shown in the image above and in the video above. For example, if the polygon has 4 vertices in the starter code, then your modified version should have 8 vertices.
2) [ 4 marks] The length of the inner vertices from the center should vary from 25 to 75. If the mouse is at the lefthand side of the canvas then the length is 25 and if the mouse is over at the righthand side then the length is 75. See the video.
3) [ 2 ] The UP_ARROW should continue to work to increase the number of sides, as shown near the end of the video.

## [ 5 marks ] Coding Style and Efficiency

Follow the course coding style for whitespace and comments. Consult the **“Code Style Guide”** on LEARN. For example:

1) [ 0.5 ] Include your name on the first line of code and your student ID number on the second line of code.
2) [ 0.5 ] Leave the third line blank.
3) [ 0.5 ] Comment your code appropriately. Avoid superfluous comments.
4) [ 0.5 ] Correctly and consistently indent your code blocks.
5) [ 0.5 ] Use correct inline spacing for variable declaration and assignment.
6) [ 0.5 ] Use good line spacing to chunk sections of your code.
7) [ 0.5 ] There are no variables that are declared or assigned, but not used.
8) [ 0.5 ] There are no unnecessary variables that are duplicates of other variables.
9) [ 0.5 ] There is no unnecessarily repeating the same code in multiple places.
10) [ 0.5 ] Semicolons were used appropriately ( i.e. at the end of most lines).

## Restrictions

- You may not use any functions or statements not covered in lecture or labs.
  This includes, but is not limited to:
  - No translate(), rotate(), or scale() functions.

# Submitting

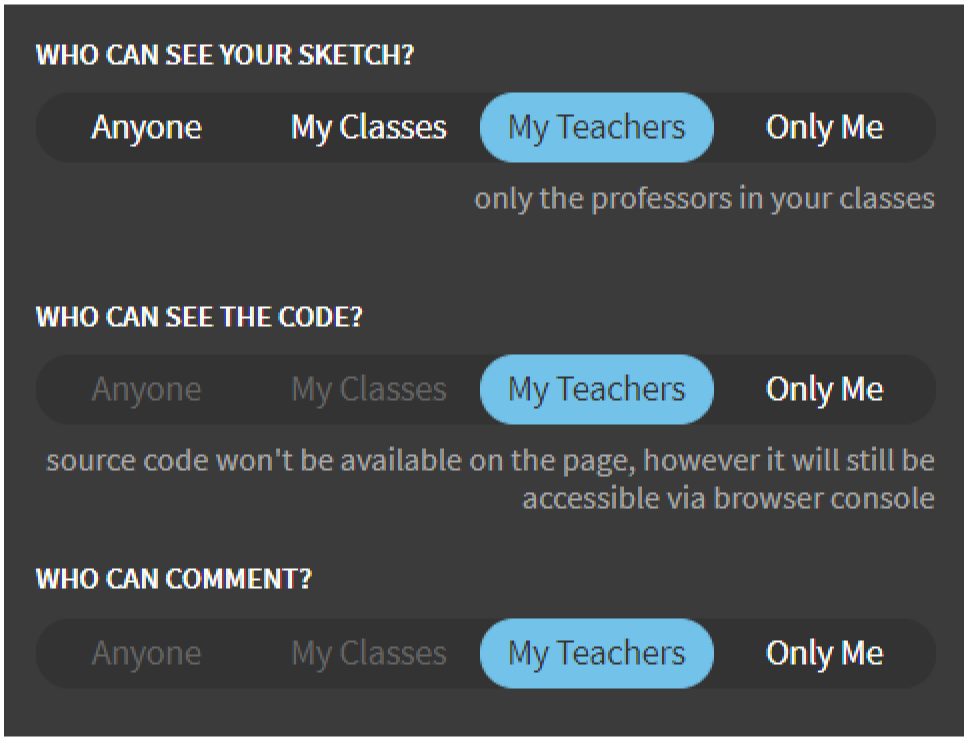
Use the template file in Word “CS106 Assignment Template” in LEARN to create your Assignment 03 submission.

Then convert your Word file to pdf. Please ensure that your URLs are hot links.  The TAs need to be able to click on each link in your pdf and go directly to your sketch.

So for example, don’t have a link like this: https://openprocessing.org/sketch/1050954

but rather have that link as a hot link as follows: [https://openprocessing.org/sketch/1050954](https://openprocessing.org/sketch/1050954)

Ensure that each URL you submit has its settings so that the access is as follows:


WHO CAN SEE YOUR SKETCH?

Anyone | My Classes | [✓ My Teachers] | Only Me

only the professors in your classes

WHO CAN SEE THE CODE?

Anyone | My Classes | [✓ My Teachers] | Only Me

source code won't be available on the page, however it will still be accessible via browser console

WHO CAN COMMENT?

Anyone | My Classes | [✓ My Teachers] | Only Me

Submit that pdf file to the Assignment 03 dropbox on LEARN.

An example of how to submit a Lab is shown in the following video:
https://vault.cs.uwaterloo.ca/s/9Xx7AGsewaea773

It is your responsibility to submit to the correct dropbox with the correct file before the deadline. Otherwise you may receive a mark of 0.

## Academic Integrity

All assignments in CS106 are done individually. Group work and sharing of code is not allowed.

Detecting Plagiarism:

- We monitor Reddit, File Trading Sites, past year CS106 assignments, etc.
- Measure Of Software Similarity (MOSS)
  - automatic system for determining the similarity of code

Discipline

- Discipline (Policy 71)
  - https://uwaterloo.ca/secretariat-general-counsel/policies-procedures-guidelines/policy-71