CPSC 217 Assignment 4

Due: Monday June 17, 2024 at 11:55pm
Weight: 7%
Sample Solution Length: 160 lines, including some comments (not including the provided code)

Individual Work:

All assignments in this course are to be completed individually. Use of large language models, such as ChatGPT, and/or other generative AI systems is prohibited. Students are advised to read the guidelines for avoiding plagiarism located on the course website. Students are also advised that electronic tools may be used to detect plagiarism.

Late Penalty:

Late assignments will not be accepted.

Submission Instructions:

Your program must be submitted electronically to the Assignment 3 drop box in D2L. It is your responsibility to ensure that your file has been uploaded successfully before the deadline. Save the confirmation email that D2L sends to you when your submission has been received successfully. You don't need to submit SimpleGraphics.py or the image files -- we already have them.

Description

A sunburst chart (sometimes referred to as a multi-level pie chart or multi-level donut chart) is a chart that can be used to effectively visualize hierarchical data. In this assignment you'll load data from a file, identify the necessary properties to draw a sunburst chart for that data, and then draw the chart.

Part 1: Loading the Data

The data for this assignment is hierarchical. The data will include some elements that are parents and other elements that are children. It is possible for an element to be both a parent and a child. This will be true if it has children itself while also being a child of another element. Such elements will be present in any chart that has more than two levels.

All of the data for this assignment is comma separated. Each line in the file will begin with the name of an element which is a parent. This will be followed by a comma and a comma separated list of all of the parent's children. Several data files can be found on the course website for this assignment, all of which are structured in this manner.

Begin by loading all of the data stored in the file into a dictionary. Each key in the dictionary will be a parent. Each value in the dictionary will be a list of the parent's children. The name of the file will be provided as a command line argument. Display the dictionary once it is loaded so that you can verify that you have stored the data correctly.

Sample output for the `emotions.txt` data file is shown below. The exact format of your output isn't particularly important, but it is critical that your keys and lists of values match what is shown below (and what is in the data files).

Emotion: ['Fear', 'Anger', 'Disgust', 'Sadness', 'Happiness', 'Surprise']

Sample output for the `canada.txt` data file is shown below:

```
ON: ['Toronto', 'Ottawa', 'Mississauga', 'Hamilton']
QC: ['Montreal', 'Quebec City']
NS: ['Halifax']
NB: ['Fredericton', 'Saint John']
PE: ['Charlottetown']
NL: ["St. John's"]
MB: ['Winnipeg']
SK: ['Regina', 'Saskatoon']
AB: ['Edmonton', 'Calgary']
BC: ['Vancouver', 'Victoria']
NU: ['Iqaluit']
NT: ['Yellowknife']
YT: ['Whitehorse']
Canada: ['NL', 'PE', 'NS', 'NB', 'QC', 'ON', 'MB', 'SK', 'AB', 'BC', 'NT', 'YT', 'NU']
```

My code for loading the data into the dictionary is approximately a dozen lines (not counting comments or blank lines).

Part 2: Is It a Leaf Element?

A leaf element is an element that does not have any children. Such elements will appear in the data dictionary in one of the values lists, but will not be a key in the dictionary.

Write a function that determines whether or not a particular element is a leaf element. Your function should take the element to examine, as well as the data dictionary as its only two parameters. The function will return True if the element is a leaf. Otherwise it will return False.

Test your function and ensure it works correctly before moving on to the next part of the assignment. All of the moons in `planets.txt` are leaf elements, as are Mercury and Venus. The Sun and all of the planets other than Mercury and Venus are not leaf elements.

My implementation of this function is less than 5 lines of code (not counting comments or blank lines).

Part 3: Find an Element's Parent

The dictionary that you created in Part 1 of the assignment maps from each parent to a list of its children, but there will be times where it is necessary to be able to identify a child element's parent element. The dictionary contains the necessary information, but it is a little bit more difficult to extract.

Write a function that takes an element and the data dictionary as it's only two parameters. Your function should return the parent element for the element that was provided as a parameter.

Test your function thoroughly before proceeding to the next part of the assignment. In `planets.txt` the parent element for all of the planets is the Sun, while the parent element for all of the moons is the planet that the moon orbits.

My implementation of this function is less than 10 lines of code (not counting comments or blank lines).

Part 4: Finding the Root Element

The root node for the chart is the one and only element in the data that is parent without also being a child. Another way of stating this is that it is the only element that is a key in the dictionary that does not appear in any of the value lists. While the root element may be the first element in the file this is not guaranteed to be the case.

Write a function that takes the dictionary storing the data for the chart as it's only parameter and returns the name of the root node as it's only result. Display the root node once you have identified it so that you can verify that you are identifying the root node correctly. Ensure that your program correctly identifies the root node for all of the provided data files.

The root node for `emotions.txt` is Emotion
The root node for `vehicles.txt` is Vehicle

My implementation of this function is less than 10 lines of code (not counting comments or blank lines).

Part 5: Counting the Number of Leaves Below an Element

Drawing the elements correctly requires the ability to determine the number of leaf elements (any number of levels) below each parent in the diagram. Computing the number of leaf elements below the root element will determine the total number of leaf elements in the chart while computing the number of leaf elements below any other element will return a smaller integer.

Create a function that counts the number of leaf nodes below a particular element. The function will take the data dictionary and the element that should have the leaf nodes below it determined as its two parameters. It will return an integer as its result. The function should return 0 if the element provided as a parameter is a leaf node.

The algorithm shown below counts the number of leaf elements below a particular element.

```
If the element for which leaves are being counted is a leaf element
    Return 0

Initialize count to 0
Initialize items to a list that contains only the element for which leaves are being counted

While items is not empty
    Remove an item from items and call it current
    If current is a leaf element
        Increment count
    Otherwise
        Add all of the children of current to the end of items

Return count
```

Be sure to test your function thoroughly. A few cases to consider include:

The number of elements below Canada in `canada.txt` is 21
The number of elements below Ontario in `candaa.txt` is 4
The number of elements below Calgary in `canada.txt` is 0

My implementation of this function is approximately a dozen lines of code (not counting comments or blank lines).

Part 6: Computing the Heights of All Elements

The height of an element is its distance from the root element. The root element has height 0. The root's children have height 1. The root's grandchildren have height 2, etc. The height of each element must be computed so that the element can be drawn in the correct location on the chart. Use the following algorithm to compute the height of each element. The algorithm takes the dictionary which maps each element to its children as its only input and returns a dictionary that maps each element to the element's height as its only result.

```
Initialize heights to an empty dictionary
Add the root element to the dictionary with height 0
Initialize items to a list that contains only the root element

While items is not empty
   Remove the first element from items and call it current
   If current is not the root element
      Find the parent element of current
      Store the current element into the heights dictionary with a value that is the height
         of parent + 1
   If current is not a leaf element
      Add all of the children of current to the beginning of items

Return heights
```

The heights of all of the elements in `shapes.txt` are shown below. Note that it's important that you have both the correct height for each element, and that the elements are in your dictionary in the order shown.

Shape: 0
2D: 1
Open: 2
Line: 3
Bezier Curve: 3
Quadratic: 4
Higher Order: 4
Cubic: 4
Closed: 2
Ellipse: 3
Polygon: 3
Triangle: 4
Quadrilateral: 4

Parallelogram: 5
Rectangle: 6
Square: 7
Trapezoid: 5
3D: 1

My implementation of this function is less than 15 lines of code (not counting comments or blank lines).

Part 7a: Drawing the Chart's Regions

This part of the assignment will guide you through drawing the sunburst chart's regions. The next part of the assignment will improve the chart by adding labels and colors. All of the functions that you have written previously will be used to perform this task so be sure that you have implemented them correctly and tested them thoroughly before proceeding.

The chart will be drawn as a collection of pie slices. These pie slices will intentionally overlap so that elements with lower heights cover part of the slice of elements with higher heights. The root element will be handled as a special case and will be drawn as a circle because the pieSlice function doesn't handle the case where the extent of the pie slice is 360 degrees in the way we would need it to for drawing this chart.

The algorithm for drawing the chart is shown below:

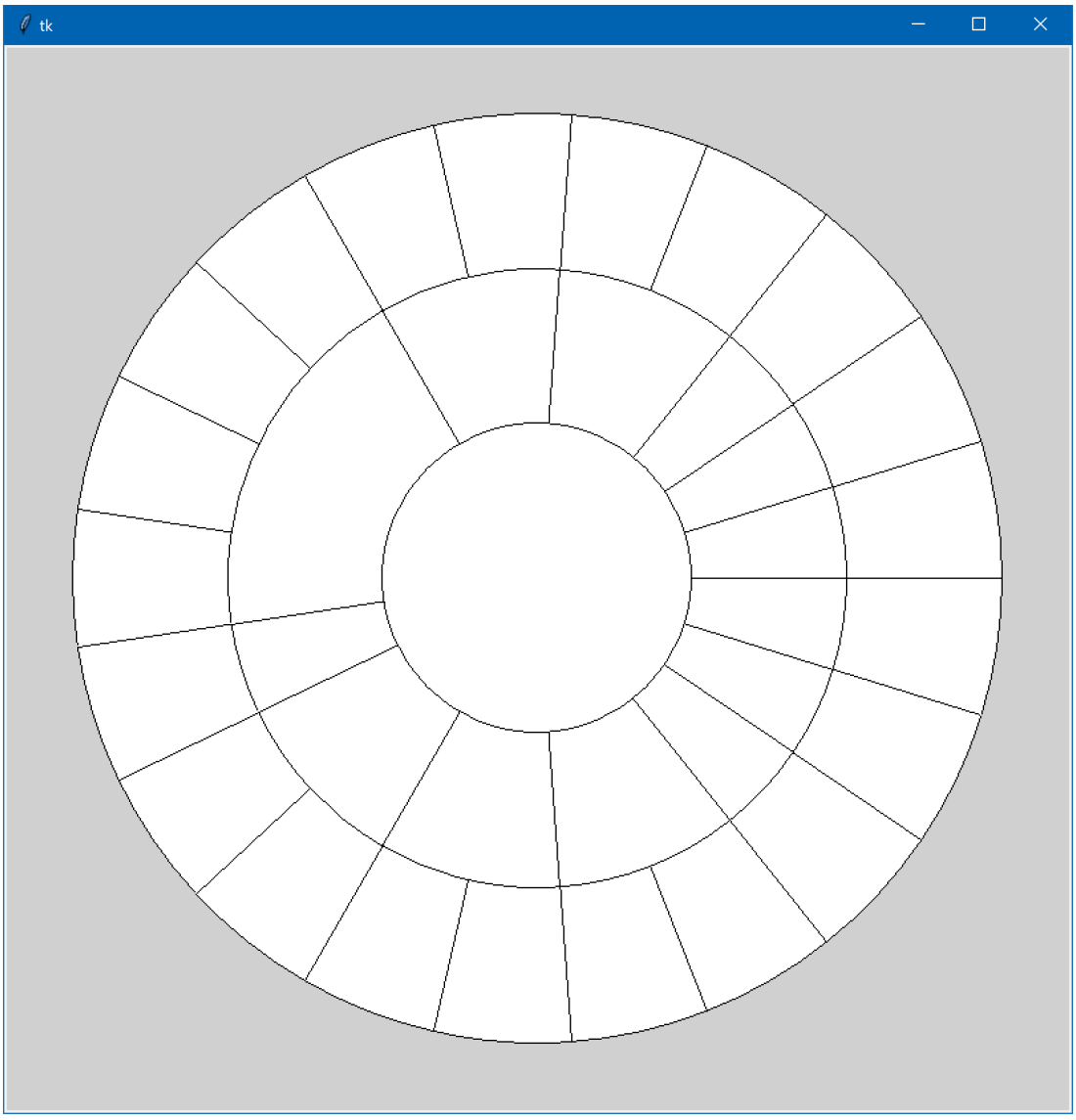
```
Identify the root node, total number of leaf elements, and the height of each element
Initialize step_size to 350 divided by (the largest height plus 1)

For each height, h, from the largest height down to (but not including) 0
   Initialize the starting angle to 0 degrees
   For each key in the heights dictionary
      If the height of the key is equal to h
         Set extent equal to 360 divided by the number of leaf elements in the chart
         If the key is not a leaf node
            Multiply extent by the number of leaf nodes below the key
         Draw a pie slice where
               • The upper left corner of the pie slice's region will have both x and y
                 coordinates that are equal to 50 + step_size * (largest height - h)
               • The width and height of the pie slice's region will be 700 - step_size *
                 (largest_height - h) * 2
               • The pie slice's starting angle will be the value currently stored in starting
                 angle
               • The pie slice's extent will be the value currently stored in extent

         Increase starting angle by extent
      Else if the height of the key is less than h and the item is a leaf item
         Set extent equal to 360 divided by the number of leaf elements in the chart
```

Increase starting angle by extent

Draw a circle for the root node where

- The upper left corner of the circle's region will have both x and y coordinates that are equal to 50 + step_size * largest height
- The width and height of the circle's region will be 700 – step_size * largest height * 2

Running your program should now draw the chart shown below for the `canada.txt` data file. My code for drawing the chart is approximately 20 lines of code.



Part 7b: Adding Labels and Color

Adding labels and color will only require some relatively minor updates to the code that you wrote in the previous part of the assignment but will significantly improve the utility and visual appeal of the chart. Begin by adding the following calculations immediately after each pie slice is drawn:

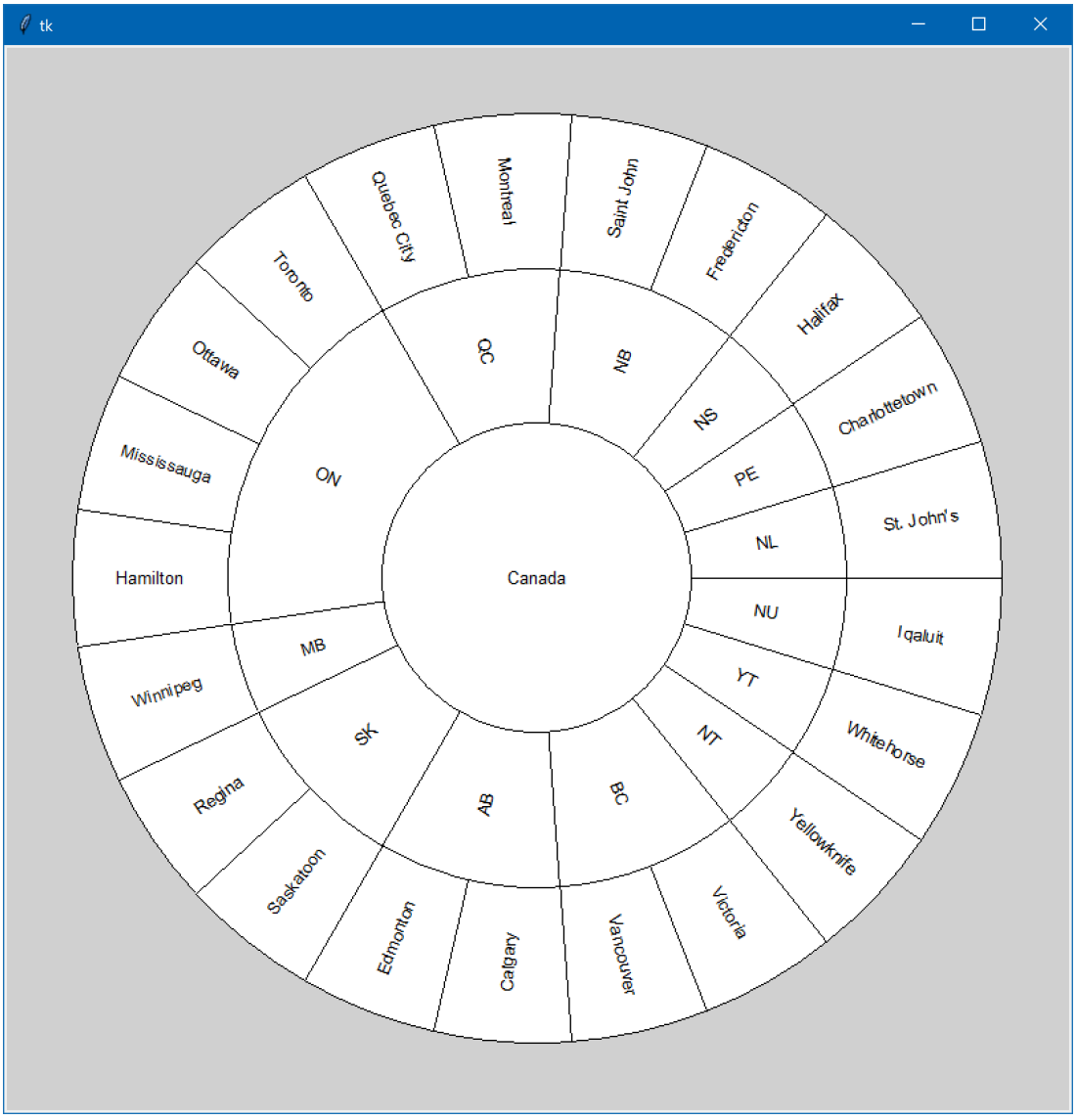
```
x = getWidth()/ 2 + (h + .5) * step_size * cos(radians(starting angle + extent / 2))
y = getHeight() / 2 - (h + .5) * step_size * sin(radians(starting angle + extent / 2))
```

(Note that you will need to modify the calculations above to use whatever variable names you chose in your program). Then use SimpleGraphics's `text` function to draw the name of each item at location (x, y).

Want to be fancy? You can, optionally, make use of the `text` function's angle argument to angle the text appropriately. If you want to be even fancier you can angle the text in the region from 90 to 270 degrees differently from the text in other areas so that the text to both the left and right of the center of the graph is upright. You could even replace spaces in the item's name with newline characters if it doesn't fit comfortably in the available space. (Nothing in this paragraph is required – it's just some ideas for things that you could explore if you are interested).

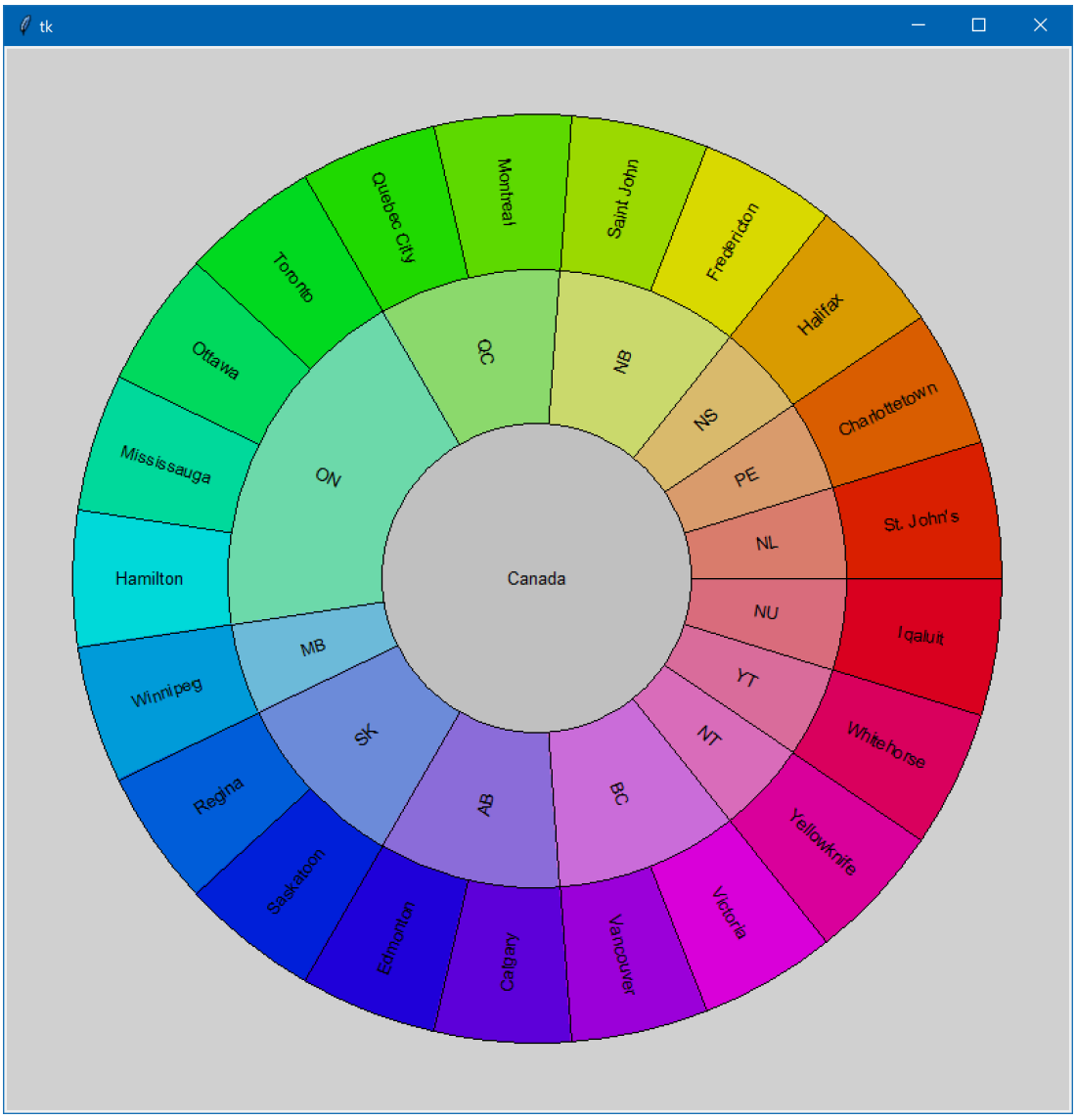
In addition, call the `text` function immediately after drawing the circle for the root node. The label for the root node should be in the center of the circle (which is also the center of the canvas).

The chart for `canada.txt` once these additions have been made is shown below.



Finally, the chart will look nicer if the regions are filled with different colors instead of all being white. The regions will be colored based on their position within the chart with colors that gradually transition as one moves around the circle and away from the root node. Such a coloring is easier to accomplish when working in the HSV color space, but SimpleGraphics expects colors to be provided in the RGB color space.

Download the hsv_to_rgb function that can be found on the course website and paste it into your assignment. (You don't need to document the source of this function). Modify your program so that it changes the fill color before each pie slice is drawn. The red, green and blue values for each slice should be computed by calling hsv_to_rgb with starting angle + extent / 2 as its first argument, h divided by the maximum height as its second argument, and 0.85 as its third argument. Set the fill color to white or light gray before drawing the circle for the root node.



Additional Requirements:

- Include your name, student ID number, and a brief description of your program at the top of your `.py` file.

- Your program must read the name of the data file from the command line. If a file name is not provided your program should either provide an appropriate error message and quit, or read the name of the file from the user.
- Your program must verify that the file provided by the user exists. If the file does not exist (or is otherwise inaccessible) your program should display an appropriate error message and quit.
- All lines of code that you write must be inside functions (except for the function definitions themselves and any constant definitions). Make appropriate use of a main function.
- Close every file that you open.
- You may assume that the data in the files you are working with is correct. Once you open the file successfully you don't need to worry about problems such as a missing comma, a blank line, or a collection of elements that doesn't represent a sunburst chart.
- You must create and use the functions described previously in this document.
- Do **not** define one function inside of another function.
- You must make appropriate use of loops. In particular, your program must work for data sets that have a variety of levels.
- Include appropriate comments on each of your functions. All of your functions should begin with a comment that briefly describes the purpose of the function, along with every parameter and every return value.
- Your program must **not** use global variables (except for constant values that are never changed).
- Your program must use good programming style. This includes things like appropriate variable names, good comments, minimizing the use of magic numbers, etc. Your program should begin with a comment that includes your name, student number, and a brief description of the program.
- Break and continue are generally considered bad form. As a result, you are **NOT** allowed to use them when creating your solution to this assignment. In general, their use can be avoided by using a combination of if statements and writing better conditions on your while loops.

Grading:

This assignment will be graded on a combination of functionality and style. A base grade will be determined from the general level of functionality of the program (Does it load and display the data? Do the functions for identifying root and leaf nodes work correctly? Can the number of leaf nodes below a node be computed? Are the heights computed correctly? Is the chart drawn correctly?). The base grade will be recorded as a mark out of 12.

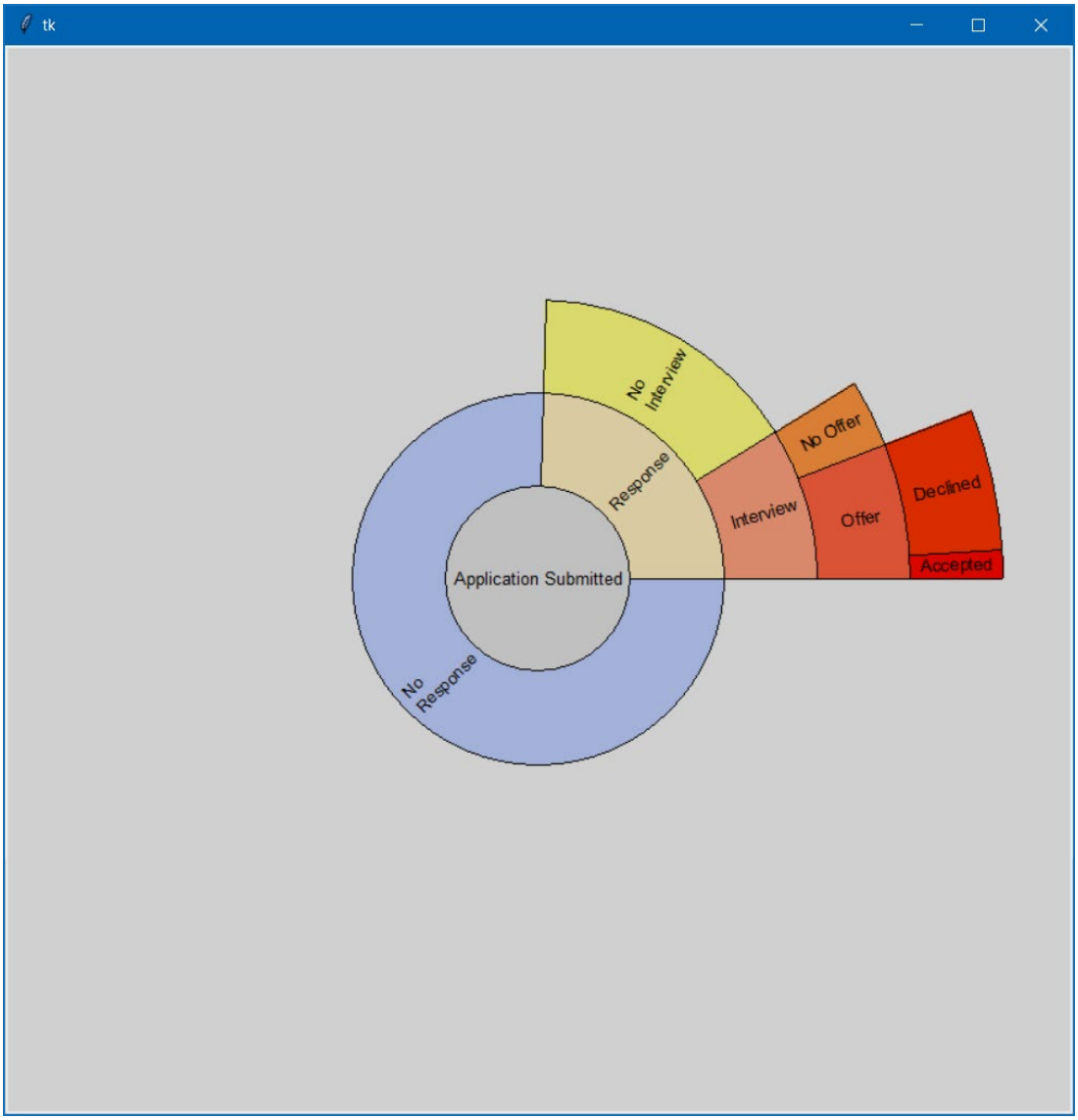
Style will be graded on a subtractive scale from 0 to -3. For example, an assignment which receives a base grade of 12 (A), but has several stylistic problems resulting in a -2 adjustment will receive an overall grade of 10 (B+). Fractional grades will be rounded to the closest integer.

Total Score (Out of 12)	Letter Grade
12	A
11	A-
10	B+
9	B

8	B-
7	C+
6	C
5	C-
4	D+
3	D
0-2	F

Looking for an A+?

A significant limitation of this program is that all of the leaf elements are the same size. In some circumstances it would be preferable to have a value associated with each leaf element and scale the size of the leaf element (and its parents) based on that value. For example, in the job search diagram it would make sense to scale the extent of each region based on the number of applications with that status, as shown below.



Update your program so that values can, optionally, be provided for the leaf nodes. If values are provided they will appear on lines with the name of the leaf node, followed by a colon, followed by the value for that leaf. You may assume that if values are provided for any leaf nodes then values will be provided for all of the leaf nodes. You may also assume that any line that includes a colon is a line providing a value for a leaf node. Ensure that your program continues to work for files that do not include values for the leaf nodes. The expected output for consoles_with_counts.txt is shown below.