CS160 Computer Science I
Program 9

Objectives
Work with lists
Work with functions
Work with files
Work with graphics

Assignment
Part 1
Write a program to ask for a series of integer numbers. Continue to ask for numbers until the user enters -1. Add each valid value to a list. Once the user has finished entering data ask for a file name. You may use `input` or `FileUtils` to ask for the file name. Write out each number to the data file, one number per line. Make sure to end each line with a newline (`"\n"`) character.

Part 2
Write each of the following functions. The function header must be implemented **exactly** as specified. Write a main function that tests each of your functions.

Specifics
In the main function ask for a filename from the user and call `fillListFromFile`. This will return a list with the values from the file. Each file should have one numeric value per line. **Do not** create this data file in this program. Then call each of the required functions and then display the results returned from the functions. Remember that the functions themselves will not display any output unless specifically stated that they should do so.

If there are built in functions in Python that will accomplish the tasks lists below YOU CANNOT USE THEM. The purpose of this lab is to get practice working with lists, not to get practice searching for methods. DO NOT sort the list, that changes the order of the items. The order of the values in the list should be the same as the order of the values in the file.

For each function that returns a numeric value determined from the list, *unless otherwise stated in the function*, if you are unable to determine a value return `None`.

Required Functions

`def fillListFromFile (fileName)` – Creates a list in the function and fills the list with the values from the `fileName` file. Then it returns the list. The file should have one number per line. No error checking is required.

`def adjustList (theList)` – Adjusts the values in `theList` such that if any value is less than 0 it is adjusted to 0, or if any value is greater than 100 it is adjusted to 100. Otherwise the value is not altered. This function returns a new list with the adjusted values. This function does not alter the original list.

`def calcAverage (theList)` – Return the average of all the values found in the list.

`def getColor (value) –` this function returns a color string based upon the value of the parameter. This function DOES NOT require a list.
Use the following table to determine the returned color:

95 – 100	90-94	70-89	60-69	0-59
darkred	red	green	blue	darkblue

`def indexOfMaxValue (theList)` – This function returns the index of the largest value in the list. This function DOES NOT return the largest value of the list. The retuned value will be in the range of 0 to the length of the list – 1. If `theList` contains the values `[3, 4, 5, 1]` this function would return 2.

def drawGraph (theList) – This function draws a vertical bar graph of the values in the list. Draw a white rectangle on a lightly colored background. I'll call this the graph window. This rectangle will contain the bar graph. For each value in the list draw a rectangle. If the value in the list is 100 the bar will reach the top of the graph window. If the value is in the list is 50 the bar will extend halfway up the graph window.

List a 1-based index for each value at the bottom of the graph.

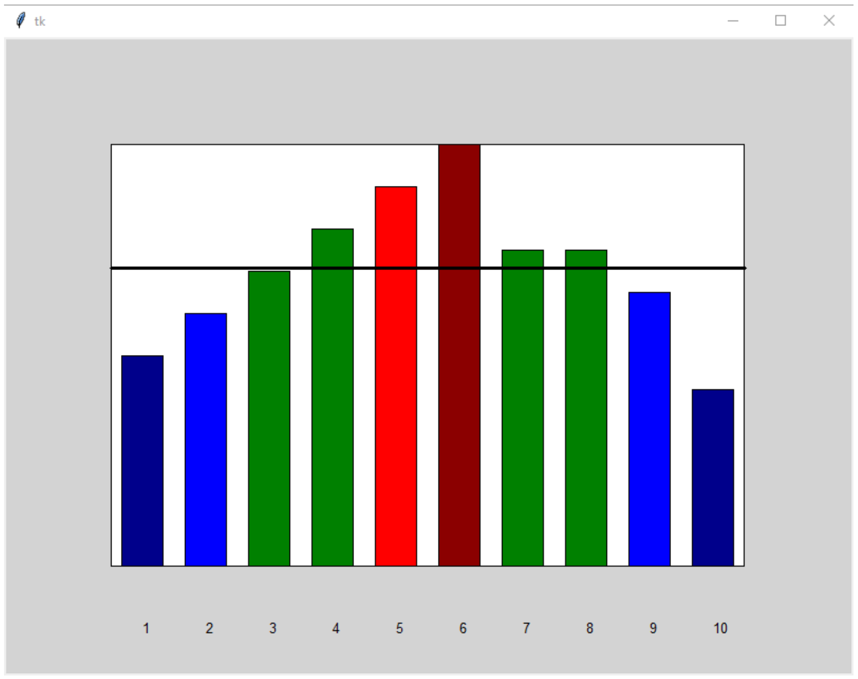
Have a horizontal bar reach across the entire graph window showing the average of values in the graph.

I used the following code to draw the graph window. Setting the variables `graphLeft`, `graphTop`, `graphWidth`, and `graphHeight` proved useful when determining the width of each bar (they will all be the same width) and the height of each individual bar. You can use your own values for the graph dimensions, this is just an example of what I did when I wrote the program.

```
from SimpleGraphics import *
...
def main():
   setSize(800, 600)
   setOutline ("black")
   setBackground ("lightgray")
   setFill ("white")
   graphLeft = 100
   graphTop = 100
   graphWidth = 600
   graphHeight = 400
   rect (graphLeft, graphTop, graphWidth, graphHeight)
...
```

For a challenge (not extra points, just extra knowledge), add 0 and 100 to the left of the graph at the appropriate heights, and the numeric average, with 2 places after the decimal point, to the side of the average line just outside of the graph. You could also add a title, with a larger font across the top of the graphics window.

Using the values of `[50, 60, 70, 80, 90, 100, 75, 75, 65, 42]`, here is an option for the graph. I added 10 pixels to the calculated x location of each bar and subtracted 20 from the width of each bar, creating the white space between the bars.



Using the values of `[55, 60, 70, 80, 90]`, here is an option for the graph.