Major Assignment 4

ENGINEER 1D04
Dr. William Farmer and Dr. Spencer Smith
McMaster University, Winter 2012

Revised: June 19, 2013

Please use AutoMarker (`automarker.mcmaster.ca`) and Avenue to acquire, test, package and submit your assignment. The procedure for submitting assignments is summarized on Avenue, with additional details provided by AutoMarker. **Please frequently back up your work by creating a submission package in AutoMarker**. This will provide a chance to recover your work in the event of an equipment failure.

Background

Let $\mathbf{R}$ be the set of real numbers, $f : \mathbf{R} \to \mathbf{R}$ be a continuous function that maps real numbers to real numbers, and $[a, b]$ be a closed interval of real numbers (i.e., a set $\{x \in \mathbf{R} \mid a \leq x \leq b\}$). The function f is *negative* on the closed interval $[a, b]$ if $f(x) < 0$ for all x in the closed interval.

Assume the function f is specified by the equation $f(x) = \mathsf{sin}(x)$ (where the input is radians). The purpose of this assignment is to write a Python program that checks whether this function is negative on a given closed interval $[a, b]$.
Design, implement, and test a program that satisfies the requirements below.

****IMPORTANT!!!**:** This assignment will be run through an automated testing program to be graded. Function syntax in your program must be **exactly** as specified, including spelling, capitalization, and the order of function parameters. **DO NOT** include a `main` function. Failure to precisely follow the requirements below will result in a **significant loss of marks**.

Requirements

1. The program contains the definition of a function named `fun` that implements f.

2. The program contains the definition of a function named `buildInputList` that takes floats a and b as input and returns a list $[x_0, \ldots, x_{100}]$, where $x_0 = min(a, b)$ and $x_{100} = max(a, b)$, as output. The x_i are evenly spaced from x_0 to x_{100} (i.e., the distance between x_i and x_{i+1} should be the same for all i with $0 \leq i \leq 99$). The members of this list are intended to serve as sample inputs for the function `fun`.

3. The program contains the definition of a function named `buildOutputList` that takes a list `[`$x_0, \ldots, x_{100}$`]` of floats as input and returns the list `[fun(`x_0`), ..., fun(`x_{100}`)]` as output.

4. The program contains the definition of a function named `checkProperty` that takes floats a and b as input and returns `True` as output if the output list shows that f is negative on each sample point in the interval $[min(a, b), max(a, b)]$ and returns `False` otherwise.

5. The program requires very little besides the function definitions. There is no `main()`.

6. The program does not read anything from standard input or write anything to standard output. That is, the program does not interact with the user who invokes it.

7. The program is written in Python in a module, NOT in the Python Shell. To create a new module in IDLE, go to File → New Window. You must save this file with a .py extension. For more information on submitting your program, click the "AutoMarker Instructions" button above.

8. Your name, MacID, student number, and the date are given in comments at the top of your Python (.py) file before your program.

9. Your answers to the design questions and the test plan (see below) are given in comments at the bottom of your Python (.py) file after your program.

10. Your program MUST have valid Python syntax and it must run without errors. Ensure that your program runs properly by running it before you submit.

11. You must sign out with a TA or IAI after you have submitted your lab at the submission station. Failure to do so could result in a zero.

Design Question

How could the `buildOutputList` function be modified so that it is more general (and thus more useful)?

Test Plan

Produce a test plan with test cases for each of the functions in your library.

```
Test:  i for function j
Input:  inputs for function j
Expected Output:  expected output for function j
```

You should have enough test cases to adequately support the argument that your code is correct. Your test cases should cover as many different classes of input cases as possible, including boundary cases. Your test plan should include case(s) where your expected output is a failure.