ECS 32B: Winter 2019
Homework Assignment 4

Due Date: No later than Saturday, February 23, 9:00pm

Submit your solutions via Canvas in one file named "hw4.py".

Here are your homework problems:

1. In the PowerPoint slides for lecture 15, you will find a function that performs binary search on a Python list using iteration. Modify that `binarySearch(alist, item)` function so that it performs ternary search on a Python list. Your new function should find two midpoints that divide the list into three (hence the name "ternary"…look it up) approximately equal size segments. If the item being searched for is found at either of the midpoints, the function should return True. If the item being searched for is less than the value at the first midpoint, ternary search continues with the segment to the "left" of the first midpoint. If the item being searched for is greater than the value at the second midpoint, ternary search continues with the segment to the "right" of the second midpoint. Otherwise, ternary search continues with the segment between the first and second midpoints. Name your function `ternarySearch`.

2. In the PowerPoint slides for lecture 15, you will find a function that performs binary search on a Python list using recursion. Modify that `binarySearchRec(alist, item)` function so that it performs ternary search on a Python list. Your new function should find two midpoints that divide the list into three approximately equal size segments. If the item being searched for is found at either of the midpoints, the function should return True. If the item being searched for is less than the value at the first midpoint, ternary search continues with the segment to the "left" of the first midpoint. If the item being searched for is greater than the value at the second midpoint, ternary search continues with the segment to the "right" of the second midpoint. Otherwise, ternary search continues with the segment between the first and second midpoints. Name your function `ternarySearchRec`.

3. The `HashTable` class implemented in chapter 5.5.3 of your online textbook (chapter 5.2.3.3 in the dead tree version) exhibits undesirable behavior if you use `put(key,val)` to add a new key-value pair when the table is full. Re-implement the `put` method so that the table will automatically increase in size when the method detects that the table is full. The new size should be the smallest prime number that is at least twice the original size. For example, if the original size is 11, the new size would be 23, not 22. You may assume that maximum size for the table is 1009 (which is the smallest prime number greater than 1000).