CS 1301 Homework 3

Homework – Next Prime & Dance
Due: Friday Feb 5th, before 6pm

THIS IS AN INDIVIDUAL ASSIGNMENT!

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

Scored out of 100 points
This is a two part homework assignment. Submit one file for each part, files must be named:
dance.py
prime.py

If you need help, we have several resources to assist you successfully complete this assignment:
- The TA Helpdesk – Schedule posted on class website.
- Email the TA's
- Jay's office hours

Notes:
• **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
• **Do not wait until the last minute to do this assignment in case you run into problems.**
• If you find a significant error in the homework assignment, please let a TA know immediately.

Part 1 – Finding the next prime number: (50 points)

Firstly, what is a prime number? A prime number, strictly speaking, is a number greater than or equal to 2 that is only divisible by 1 and itself (0 can be divided by infinitely many numbers, and 1 can only be divided by itself. Negative numbers are typically not included). For example, the first few prime numbers are 2, 3, 5, 7, 11, 13, and 17.

isPrime

Write a function named **isPrime(num)** that accepts a single argument (num) and **returns** True if the argument is prime, and False otherwise. You may assume that isPrime will only be tested with numbers up to 10,000,019. Your function MUST WORK for larger numbers, even if it would take a long time, but don't worry about making it run super fast, we will only test with a few numbers. As long as it works on 10,000,019 within 1 minute you are fine.

Examples: isPrime(2) should return True. isPrime(5) should return True as well. isPrime(4) should return False. isPrime(2000003) should return True. You may assume that the input **num** will be a non-negative integer. (0, 1, 2, etc...)

nextPrime

Next, write a second function named **nextPrime(num)** that will *return* the next prime number following the corresponding number that is the argument. It is likely that your **nextPrime(num)** function can make use of the **isPrime(num)** function to make it's job easy! You may assume that nextprime will only be tested with numbers up to 10,000,000 (Ten million). Your function MUST WORK for larger numbers, even if it would take a long time, but don't worry about making it run super fast, as long as it can work for 10,000,000 within 3 minutes you are fine.

For example: nextPrime(5) should *return* (NOT PRINT) the integer 7, nextPrime(100) should *return* the integer 101, and nextPrime(10000000) should return the number 1000019. nextPrime(0) should return 2. You may assume that only non-negative integers will be used as input.

Save and submit your two functions in a single file called **prime.py**

Grading Criteria for Part 1:

isPrime

Shows correct use of conditionals and loops:	-	10 pt
Produces correct results for all inputs (including 0 and 1):	-	10 pt
Function named correctly and returns a boolean (instead of printing)	-	5 pt

nextPrime

Shows correct use of conditionals and loops:	-	10 pt
Produces correct results for all inputs	-	10pt
Function named correctly and returns an int (instead of printing)	-	5 pt

NextPrime Written By: Jason Tilley, Fall 2009

Part 2 – Dance, Robot, Dance! (50 points)

Hopefully by now, you’ve gotten your robot out of the box and made friends with it (or at least acquaintances); given it a name and a long, intricate history. Well, how about now you take it out to a dance?

Of course, you’ll have to teach your robot to dance. Using the movement functions:

http://wiki.roboteducation.org/Myro_Reference_Manual#Movement_Functions

have your robot do a little dance. The dance should last for at least 30 seconds, and contain at least 3 distinct dance moves. Don’t just go back and forth for 30 seconds; vary the dance a bit. Pretend your robot is well-versed in rhythm and soul, or is at least a little spastic. In addition to the movement, your robot should also make some noise while it's moving! The **beep()** function is very helpful – it allows the robot produce various tones.

To make your robot move while at the same time beeping, you will need to take advantage of the asynchronous movement function calls. For example, look at the difference in behavior of the following two code snippets:

Using Synchronous movement call:	Using Asynchronous movement call:

`def danceMove1():`
`   forward(1,1)`
`   beep(1,440)`
	`def danceMove2():`
	`   forward(1)`
	`   beep(1,440)`
	`   stop()`

You are allowed (and encouraged) to make your own helper functions that contain individual dance moves, and then call these helper functions repeatedly from a main function called dance(), using loops.

Write your dance as a function called **dance()**, and save it into a file called **dance.py**. As always, please name your file exactly as requested.

Grading Criteria for Part 2 – Dance

Function named correctly (dance)	-	5 points
File named correctly (dance.py)	-	5 points
Dance lasts for at least 30 seconds	-	5 points
Contains at least 3 distinct moves	-	15 points
Robot moves while (at the same time!) making a sound	-	15 points
Creativity: Dance is interesting!	-	5 points

Total for part 2: 50 points

Written By: Melody Nailor, Spring 2009; Kevin Jones, Summer 2009.