CSc 210: Software Development
Section Sol 1: Welcome back (to programming)!
August 28th, 2017

Name: ______________________ Netid: ______________________

The purpose of these problems is to refresh your memory on key concepts you should remember from CS 110/120 as well as to get you working on the command line.

Problem 1

A palindrome is a word that is the same forward and backward. Example: “racecar” is spelled the same both forward and backward.

Write a recursive function called is_pal(str, pos1, pos2) to discover if a given string is a palindrome. The program should take the word it is performing the operation on from the command line. Given the following commands, your program should produce the shown output:

```
% python3 is_pal.py racecar
True
% python3 is_pal.py not_a_palindrome
False
```

```
def main():
    s1 = sys.argv[1]
    print(pal(s1, 0, len(s1)- 1))


def pal(s, pos1, pos2):

    if pos1 > pos2:
        return True
    elif s[pos1] == s[pos2]:
        return pal(s, pos1 + 1, pos2 - 1)
    else:
        return False

main()
```

Problem 2

Implement a python program called isort.py that reads in a stream of integers from standard input. It should sort this integers using insertion sort, and print them out. It should behave as shown below.

```
% python3 isort.py
3
1
2
^D (CTRL-D, end of input)
1 2 3
```

```
import sys

def main():

    nums = []
    for line in sys.stdin:
        for num in line.split():
            nums.append(num)

    nums = isort(nums)
    print(nums)

def isort(arr):
    for i in range(1,len(arr)):
        j = i
        while j > 0 and arr[j] < arr[j-1]:
            arr[j], arr[j-1] = arr[j-1], arr[j]
            j=j-1
    return arr

main()
```

Problem 3

Recall the concept of a binary tree. Essentially, we have nodes that hold data inside of them and each node has two children associated with it which also hold data. Recall that a binary search tree is defined as the data in the left child of a parent node to be considered respectively less than the data in the parent and the data in the right child should be considered respectively greater than that of the parent. In order for a binary tree to be a BST, this invariant must hold at all nodes. Also recall we can perform different traversals on our tree such as inorder, preorder, or postorder traversals.

Imagine we have the following binary tree node class:

```
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
    def set_left(self, node):
        self.left = node
    def set_right(self, node):
        self.right = node
```

For this problem, write a recursive function called is_bst(node) that determines if a binary tree maintains the BST invariant by doing one of the mentioned traversals on it.

```
def is_bst(node):

    if (node.left != None):
        if not is_bst(node.left):
            return False
        if (node.left.data > node.data):
            return False


    if (node.right != None):
        if not is_bst(node.right):
            return False
        if node.right.data < node.data:
            return False

    return True
```