CSc 110 Midterm 2 Sample Exam #1

1. List Mystery

Consider the following function:

```
def list_mystery(list):
    x = 0
    for i in range(len(list) - 1):
        if list[i] > list[i + 1]:
            x += 1
    return x
```

In the left-hand column below are specific lists of integers. Indicate in the right-hand column what value would be returned by function `list_mystery` if the integer list in the left-hand column is passed as its parameter.

Original Contents of List	Value Returned
`a1 = [8]` `result1 = list_mystery(a1)`	_____
`a2 = [14, 7]` `result2 = list_mystery(a2)`	_____
`a3 = [7, 1, 3, 2, 0, 4]` `result3 = list_mystery(a3)`	_____
`a4 = [10, 8, 9, 5, 6]` `result4 = list_mystery(a4)`	_____
`a5 = [8, 10, 8, 6, 4, 2]` `result5 = list_mystery(a5)`	_____

2. While Loop Mystery

For each call of the function below, write the output that is printed:

```
def mystery(i, j):
    while i != 0 and j != 0:
        i = i // j
        j = (j - 1) // 2
        print(str(i) + " " + str(j) + " ", end='')
    print(i)
```

Function Call	Output
`mystery(5, 0)`	_____
`mystery(3, 2)`	_____
`mystery(16, 5)`	_____
`mystery(80, 9)`	_____
`mystery(1600, 40)`	_____

3. Assertions

For the following function, identify each of the three assertions in the table below as being either ALWAYS true, NEVER true or SOMETIMES true / sometimes false at each labeled point in the code. You may abbreviate these choices as A/N/S respectively.

```
def mystery():
    y = 0
    z = 1
    next = input()

    # Point A
    while next >= 0:
        # Point B
        if y > z:
            # Point C
            z = y
        y += 1
        next = input()
        # Point D

    # Point E
    return z
```

	next < 0	y > z	y == 0
Point A			
Point B			
Point C			
Point D			
Point E			

4. File Processing

Write a function named `word_stats` that accepts as its parameter the name of a file that contains a sequence of words and that reports the total number of words (as an integer) and the average word length (as an un-rounded real number). For example, suppose file contains the following words:

```
To be or not to be, that is the question.
```

For the purposes of this problem, we will use whitespace to separate words. That means that some words include punctuation, as in `"be,"`. For the input above, your function should produce exactly the following output:

```
Total words    = 10
Average length = 3.2
```

6. List Programming

Write a function named `min_gap` that accepts an integer list as a parameter and returns the minimum 'gap' between adjacent values in the list. The gap between two adjacent values in a list is defined as the second value minus the first value. For example, suppose a variable called `list` is a list of integers that stores the following sequence of values.

```
list = [1, 3, 6, 7, 12]
```

The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7). Thus, the call of `min_gap(list)` should return `1` because that is the smallest gap in the list. Notice that the minimum gap could be a negative number. For example, if list stores the following sequence of values:

```
[3, 5, 11, 4, 8]
```

The gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned.

This gap information can be helpful for determining other properties of the list. For example, if the minimum gap is greater than or equal to 0, then you know the array is in sorted (nondecreasing) order. If the gap is greater than 0, then you know the list is both sorted and unique (strictly increasing).

If you are passed an list with fewer than 2 elements, you should return 0.

6. Programming

Write a function named `longest_sorted_sequence` that accepts a list of integers as a parameter and that returns the length of the longest sorted (nondecreasing) sequence of integers in the list. For example, if a variable named `list` stores the following values:

```
List = [3, 8, 10, 1, 9, 14, -3, 0, 14, 207, 56, 98, 12]
```

then the call of `longest_sorted_sequence(list)` should return `4` because the longest sorted sequence in the array has four values in it (the sequence -3, 0, 14, 207). Notice that sorted means nondecreasing, which means that the sequence could contain duplicates. For example, if the list stores the following values:

```
list2 = [17, 42, 3, 5, 5, 5, 8, 2, 4, 6, 1, 19]
```

Then the function would return `5` for the length of the longest sequence (the sequence 3, 5, 5, 5, 8). Your function should return 0 if passed an empty list. Your function should return 1 if passed a list that is entirely in decreasing order or contains only one element.

Solutions

1.

Call	Value Returned
`a1 = [8]` `result1 = list_mystery (a1)`	`0`
`a2 = [14, 7]` `result2 = list_mystery (a2)`	`1`
`a3 = [7, 1, 3, 2, 0, 4]` `result3 = list_mystery (a3)`	`3`
`a4 = [10, 8, 9, 5, 6]` `result4 = list_mystery (a4)`	`2`
`a5 = [8, 10, 8, 6, 4, 2]` `result5 = list_mystery (a5)`	`4`

2.

Function Call	Output
`mystery(5, 0)`	`5`
`mystery(3, 2)`	`1 0 1`
`mystery(16, 5)`	`3 2 1 0 1`
`mystery(80, 9)`	`8 4 2 1 2 0 2`
`mystery(1600, 40)`	`40 19 2 9 0 4 0`

3.

	next < 0	y > z	y == 0
Point A	SOMETIMES	NEVER	ALWAYS
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	NEVER	ALWAYS	NEVER
Point D	SOMETIMES	SOMETIMES	NEVER
Point E	ALWAYS	SOMETIMES	SOMETIMES

4.

```
def word_stats(file_name):
    words = open(file_name).read().split()
    count = 0
    sum_length = 0

    for word in words:
        count += 1
        sum_length += len(word)

    average = sum_length / count
    print("Total words    = " + str(count))
    print("Average length = " + str(average))
```

6.

```
def min_gap(list):
    if len(list) < 2:
        return 0
    else:
        min = list[1] - list[0]
        for i in range(2, len(list)):
            gap = list[i] - list[i - 1]
            if gap < min:
                min = gap
        return min
```

7.

```
def longest_sorted_sequence(list):
    if len(list) == 0:
        return 0

    max = 1
    count = 1
    for i in range(1, len(list)):
        if list[i] >= list[i - 1]):
            count += 1
        else:
            count = 1

        if count > max:
            max = count
    return max
```