CSc 110 Midterm 2 Sample Exam #2

1. List Mystery

Consider the following function:

```
def list_mystery(a):
    for i in range(len(a) - 2, 0, -1):
        if a[i + 1] <= a[i - 1]:
            a[i] += 1
```

Indicate in the right-hand column what values would be stored in the list after the function list_mystery executes if the integer list in the left-hand column is passed as a parameter to it.

Original Contents of List	Final Contents of List

`a1 = [42]`
`list_mystery(a1)`
	________________________________

`a2 = [1, 8, 3, 6]`
`list_mystery(a2)`
	________________________________

`a3 = [5, 5, 5, 5, 5]`
`list_mystery(a3)`
	________________________________

`a4 = [10, 7, 9, 6, 8, 5]`
`list_mystery(a4)`
	________________________________

`a5 = [1, 0, 1, 0, 0, 1, 0]`
`list_mystery(a5)`
	________________________________

2. While Loop Mystery

For each call of the method below, write the value that is returned:

```
def mystery(i, j):
    k = 0
    while i > j:
        i = i - j
        k = k + (i - 1)
    return k
```

```
Function Call                                   Value Returned

mystery(2, 9)                                   ______________________________

mystery(5, 1)                                   ______________________________

mystery(38, 5)                                  ______________________________

mystery(5, 5)                                   ______________________________

mystery(40, 10)                                 ______________________________
```

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 assertions(n):
    x = 2

    # Point A
    while x < n:
        # Point B
        if n % x == 0:
            n = n // x
            x = 2
            # Point C
        Else:
            x += 1
            # Point D

    # Point E
    return n
```

	`x > 2`	`x < n`	`n % x == 0`
Point A			
Point B			
Point C			
Point D			
Point E			

4. File Processing

Write a function named `find_first_match` that accepts as its parameters an input file name and a list of `String` *keywords* representing a list of keywords in a search. Your function will read lines from the input file and should return the line number of the first line in the file that contains one or more words from *keywords*. If none of the keywords are found in the file, your function should return a -1. The search should be case-insensitive, so if a keyword was "banana", the line "yummy baNAna split" would be considered a line that contains the keyword. Your function should also match **whole words** only, so if the only keyword in the list was "ball", the line "football game" would *not* be considered a match.

For example, consider the following input file saved in `sidewalk.txt`, consisting of 6 lines:

```
Let us leave this place where the smoke blows black
And the dark street winds and bends.
Past the pits where the asphalt flowers grow
We shall walk with a walk that is measured and slow,
And watch where the chalk-white arrows go
To the place where the sidewalk ends.
```

The following table shows some calls to your function and their expected results.

List	Call / Returned Value
`k1 = ["place", "winds"]`	`find_first_match("sidewalk.txt", k1)` returns `1`
`k2 = ["dinosaur", "PITS", "pots"]`	`find_first_match("sidewalk.txt", k2)` returns `3`
`k3 = ["chalk", "row", "g", "ends"]`	`find_first_match("sidewalk.txt", k3)` returns `-1`
`k4 = ["to"]`	`find_first_match("sidewalk.txt", k4)` returns `6`

You may assume that none of the words in the *keywords* list contain spaces, i.e. all keywords are single whole words, and the list contains at least one element. Do not modify the elements of the *keywords* list.

5. List Programming

Write a function named `zero_out` that accepts two lists of integers *a1* and *a2* as parameters and replaces any occurrences of *a2* in *a1* with zeroes. The sequence of elements in *a2* may appear anywhere in *a1* but must appear consecutively and in the same order. For example, if variables called `a1` and `a2` store the following values:

```
a1 = [1, 2, 3, 4, 1, 2, 3, 4, 5]
a2 = [2, 3, 4]
```

The call of `zero_out(a1, a2)` should modify `a1`'s contents to be `[1, 0, 0, 0, 1, 0, 0, 0, 5]`.
Note that the pattern can occur many times, even consecutively. For the following two lists `a3` and `a4`:

```
a3 = [5, 5, 5, 18, 5, 42, 5, 5, 5, 5]
a4 = [5, 5]
```

The call of `zero_out(a3, a4)` should modify `a3`'s contents to be `[0, 0, 5, 18, 5, 42, 0, 0, 0, 0]`.

You may assume that both lists passed to your function will have lengths of at least 1. If *a2* is not found in *a1*, or if *a1*'s length is shorter than *a2*'s, then *a1* is not modified by the call to your function. Please note that *a1*'s contents are being modified in place; you are not supposed to return a new list. Do not modify the contents of *a2*.

6. Programming

Write a function `coin_flip` that accepts as its parameter an input file name. Assume that the input file data represents results of sets of coin flips that are either heads (H) or tails (T) in either upper or lower case, separated by at least one space. Your function should consider each line to be a separate set of coin flips and should output to the console the number of heads and the percentage of heads in that line, rounded to the nearest tenth. If this percentage is more than 50%, you should print a "You win" message. For example, consider the following input file:

```
H T H H T
T t   t T h   H
   h
```

For the input above, your function should produce the following output:

```
3 heads (60.0%)
You win!

2 heads (33.3%)

1 heads (100.0%)
You win!
```

The format of your output must exactly match that shown above. You may assume that input file contains at least 1 line of input, that each line contains at least one token, and that no tokens other than h, H, t, or T will be in the lines.

Solutions

1. List Mystery

Call	Final Contents of List

`a1 = [42]`
`list_mystery(a1)`
	`[42]`

`a2 = [1, 8, 3, 6]`
`list_mystery(a2)`
	`[1, 8, 4, 6]`

`a3 = [5, 5, 5, 5, 5]`
`list_mystery(a3)`
	`[5, 6, 5, 6, 5]`

`a4 = [10, 7, 9, 6, 8, 5]`
`list_mystery(a4)`
	`[10, 8, 10, 7, 9, 5]`

`a5 = [1, 0, 1, 0, 0, 1, 0]`
`list_mystery(a5)`
	`[1, 1, 1, 1, 0, 2, 0]`

2. While Loop Mystery

Function Call	Value Returned
`mystery(2, 9)`	0
`mystery(5, 1)`	6
`mystery(38, 5)`	119
`mystery(5, 5)`	0
`mystery(40, 10)`	57

3. Assertion

	`x > 2`	`x < n`	`n % x == 0`
Point A	NEVER	SOMETIMES	SOMETIMES
Point B	SOMETIMES	ALWAYS	SOMETIMES
Point C	NEVER	SOMETIMES	SOMETIMES
Point D	ALWAYS	SOMETIMES	SOMETIMES
Point E	SOMETIMES	NEVER	SOMETIMES

4. File Processing

```
def find_first_match(file_name, keywords):
    lines = open(file_name).readlines()
    line_num = 0
    for line in lines:
        words = line.split()
        line_num += 1
        for word in words:
            for i in range(0, len(keywords)):
                if keywords[i].lower() == word.lower():
                    return line_num
    return -1
```

5. List Programming

```
def zero_out(a1, a2):
    for i in range(0, len(a1) - len(a2)):
        count = 0
        for j in range(0, len(a2)):
            if a1[i + j] == a2[j]:
                count += 1

        if count == len(a2):  # found it
            for j in range(0, len(a2)):
                a1[i + j] = 0
```

6. Programming

```
def coin_flip(file_name):
    lines = open(file_name).readlines()
    for line in lines:
        upper_line = line.upper()
        heads = 0
        total = 0
        for flip in upper_line:
            total += 1
            if flip == "H":
                heads += 1

        percent = 100 * heads / total
        rounded = round(10 * percent) / 10
        print(str(heads)+ " heads (" + str(rounded) + "%)")
        if percent > 50:
            print("You win!")
        print()
```