Midterm Exam

● Graded

Student

Evan O'Neill

Total Points

90 / 100 pts

Question 1

Q1

6 / 6 pts

✔ **– 0 pts** Correct

– 3 pts Missing definition of "unambiguous"

– 2 pts Incomplete definition of "unambiguous"

– 3 pts Missing example

– 2 pts Incomplete or incorrect example

– 6 pts No answer

Question 2

Q2

6 / 6 pts

✔ **– 0 pts** Correct

– 3 pts Missing definition of Modular Design

– 2 pts Incomplete or incorrect definition of Modular Design

– 3 pts Missing example

– 2 pts Incomplete or incorrect example

– 6 pts No answer

Question 3

Q3

5 / 6 pts

- **– 0 pts** Correct
- **– 2 pts** i = 3 -- missing
- **– 1 pt** i incorrectly initialized
- **– 2 pts** while i < n: -- missing
- ✔ **– 1 pt** Incorrect while condition
- **– 2 pts** i = i + 1 -- missing
- **– 1 pt** Incomplete increment of counter
- **– 6 pts** No answer

Question 4

Q4

6 / 6 pts

4.1 Q4a

3 / 3 pts

- ✔ **– 0 pts** Correct
- **– 3 pts** Missing answer to part a
- **– 3 pts** Incorrect answer to part a
- **– 2 pts** Incomplete answer to part a

4.2 Q4b

3 / 3 pts

- ✔ **– 0 pts** Correct
- **– 3 pts** Missing answer to 4b
- **– 3 pts** Incorrect answer to 4b
- **– 2 pts** Incomplete answer to 4b

Question 5

Q5 **8** / 8 pts

5.1 **Q5a** **4** / 4 pts

- ✔ **– 0 pts** Correct
- **– 4 pts** No tracing shown on paper
- **– 2 pts** Incorrect or incomplete tracing shown on paper
- **– 1 pt** program is called once, no need for more tracing

5.2 **Q5b** **4** / 4 pts

- ✔ **– 0 pts** Correct
- **– 4 pts** Missing or incorrect answer for output
- **– 1 pt** Missing output below

Question 6

Q6 **8** / 8 pts

6.1 **Q6a** **4** / 4 pts

- ✔ **– 0 pts** Correct
- **– 4 pts** No tracing shown on paper
- **– 2 pts** Incorrect or incomplete tracing shown on paper

6.2 **Q6b** **4** / 4 pts

- ✔ **– 0 pts** Correct
- **– 4 pts** Missing output
- **– 2 pts** Value for b incorrect
- **– 2 pts** Value for i incorrect
- **– 2 pts** outputs partially written wrong

Question 7

Q7

3 / 10 pts

- **− 0 pts** Correct
- **− 9 pts** No tracing shown for any functions
- **− 3 pts** No tracing shown for fun1
- ✔ **− 2 pts** Incomplete or incorrect tracing shown for fun1
- **− 3 pts** No tracing shown for fun2
- ✔ **− 2 pts** Incomplete for incorrect tracing shown for fun2
- **− 3 pts** No tracing shown for main
- ✔ **− 2 pts** Incomplete or incorrect tracing shown for main
- ✔ **− 1 pt** Missing or incorrect ouput
- **− 10 pts** No answer provided

Question 8

Q8

10 / 10 pts

8.1 **Q8a**

6 / 6 pts

- ✔ **− 0 pts** Correct
- **− 2 pts** No answer or incorrect answer for 'bill'
- **− 2 pts** No answer or incorrect answer for 'bob'
- **− 2 pts** No answer or incorrect answer for '1234'
- **− 6 pts** No answer provided for 8a
- **− 1 pt** need to write as a string not as a list
- **− 1 pt** too disorganized

8.2 **Q8b**

4 / 4 pts

- ✔ **− 0 pts** Correct
- **− 4 pts** No answer for part b
- **− 2 pts** Incomplete or incorrect answer for part b

Question 9

Q9 **8** / 10 pts

- **− 0 pts** Correct
- **− 2 pts** Count never initialized or initialized incorrectly
- **− 4 pts** Incorrect loop setup
- **− 2 pts** incorrect start point for the loop
- ✔ **− 2 pts** incorrect end point for the loop
- **− 1 pt** incorrect or missing comparison
- **− 2 pts** Count never incremented
- **− 2 pts** infinite loop
- **− 10 pts** No answer provided
- **− 2 pts** Unnecessary change of parameter values

Question 10

Q10 **10** / 10 pts

- ✔ **− 0 pts** Correct
- **− 2 pts** countList = [] -- missing
- **− 2 pts** for ch in str1 OR for i in range(len(str1)) -- missing
- **− 4 pts** missing call to countChar
- **− 2 pts** incorrect call to countChar
- **− 2 pts** missing append to countList
- **− 10 pts** No answer provided

Question 11

Q11 **10** / 10 pts

11.1 Q11a **5** / 5 pts

✔ **– 0 pts** Correct

– 5 pts No answer provided / insufficient answer

GetData

– 0.5 pts Missing or incorrect loop setup

– 0.5 pts missing or incorrect file open

– 0.5 pts missing or incorrect file close

– 0.5 pts too many or too few lines being read in

– 0.5 pts Missing or incorrect line strip

– 0.5 pts missing or incorrect line split

– 1 pt Both appends missing or incorrect

– 0.5 pts Inventory data should be taken in as an int

---

findLowInventory

– 1 pt Missing or incorrect loop setup

– 1 pt Incorrect or missing comparison

– 1 pt Incorrect append

– 1 pt Threshold should not be asked for inside of the function

---

countAllToys

– 1 pt incorrect loop setup

– 3 pts Missing read, strip, split, open

11.2 **Q11b** **5** / 5 pts

✔ **– 0 pts** Correct

– 5 pts No answer provided / insufficient answer

GetData

– 0.5 pts Missing or incorrect loop setup

– 0.5 pts missing or incorrect file open

– 0.5 pts missing or incorrect file close

– 0.5 pts too many or too few lines being read in

– 0.5 pts Missing or incorrect line strip

– 0.5 pts missing or incorrect line split

– 1 pt Both appends missing or incorrect

– 0.5 pts Inventory data should be taken in as an int

---

findLowInventory

– 1 pt Missing or incorrect loop setup

– 1 pt Incorrect or missing comparison

– 1 pt Incorrect append

– 1 pt Threshold should not be redefined inside of the function

– 1 pt Missing or incorrect return statement

CountAllToys

– 1 pt Incorrect loop setup

– 1 pt Unnecessary comparison

– 1 pt accumulator never initialized

– 1 pt Incorrect way to add to the accumulator

– 1 pt Missing or incorrect return statement

– 3 pts Missing loop, threshold is a parameter

Question 12

Q12

10 / 10 pts

✔ **– 0 pts** Correct

– 2 pts Missing call to getData

– 1 pt Incorrect call to getData

– 2 pts Missing get threshold from user

– 0.5 pts Threshold should be an int

– 2 pts Missing call to findLowInventory

– 1 pt Incorrect call to findLowInventory

– 2 pts Missing call to countAllToys

– 1 pt Incorrect call to countAllToys

– 2 pts Missing call to printResults

– 1 pt Incorrect call to printResults

– 10 pts No answer provided

as a note, 'etoys.csv' should be wrapped in quotations as it's a string.

CSC 110 – Midterm Exam
October 21, 2022

Name: ~~Evan O'Neill~~ Evan O'Neill

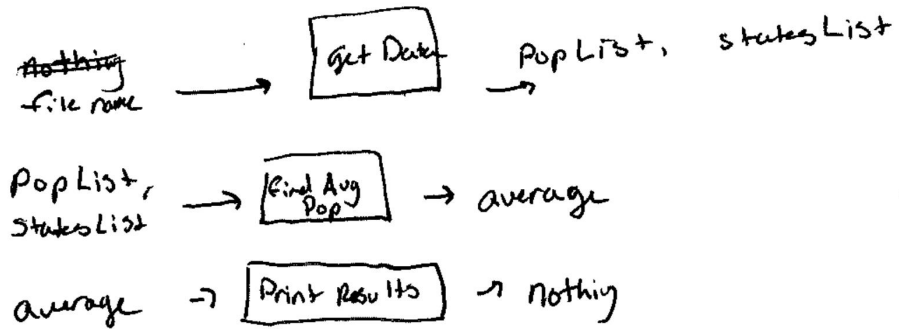
1. In the definition of an algorithm that we discussed in class what is meant when we say that an operation must be *unambiguous*? Give an example, using python code, of a set of operations that is not unambiguous. (6 pts)

Unambiguous means that the computer is given clear directions to execute.

```
i = 3
j = 4
While  i >=3 :
    j = j - 1
    i = i - 1
    k = i + 4
print("The number is", k)
```

2. When using functions in a python program, what do we mean by Modular Design? Show how you would apply modular design to solve the following problem: Given a data file containing states and populations, find the average population across all states. You SHOULD NOT write code here. You can use diagrams instead. (6 pts)

Modular Design means breaking a larger problem into smaller problems, and solving the smaller problems first.



3. Modify the following code so that it uses a while loop instead of a for loop. (6pts)

```
n = 10
numList = []
for i in range(3,n):
      x = i % 3
      numList.append(x)
print(numList)
```

```
numList = []
n = 10
i = 3
while i=<n:
    x = i %o 3
    numList = numList + [x]
    i = i + 1
```

4. Given the following parallel lists of snack items and the number of calories per serving for the snack, write a few python statements to do the following: (6pts)

```
snackList = ['Oreos', 'Snickers', 'Thin Mint', 'Ritz Bits', 'Skittles']
calorieList = [471, 215, 161, 160, 156]
```

a) Print the number of calories in a serving of Ritz Bits. Assume that you don't know where 'Ritz Bits' is in the list and you have to find it.

```
for i in range(len(snackList)):
    if snackList[i] == 'Ritz Bits':
        print('Ritz Bits has', calorieList[i], 'calories.')
```

b) Add the following information to the lists: 'Almond Joy' has 234 calories.

```
snackList = snackList + ['Almond Joy']
calorieList = calorieList + [234]
```

$$12 \overline{)42} \quad 3, \quad -36, \quad 6$$

5. What is the output of the following code? Use the space on the right to trace the code. You must show your work in the trace to receive full credit. (8pts)

```
x = 42
z = 11
y = x % (z+1)
if z >= y:
    print('Apple')
elif z < y or y < 20:
    print('Pear')
else:
    print('Grape')
```

x	z	y
42	11	6

11 >= 6 ✓

Output: "Apple" Apple

6. What is the output of the following code? Use the space on the right to trace the code. You must show your work in the trace to receive full credit. (8 pts)

```
b = 16
i = 24
while i > b:
    b = b + 2
    i = i - 3
print("b: ", b)
print("i: ", i)
```

b	i
16	24
18	21
20	~~18~~ 18

Output:

b: 20

i: ~~18~~ 18

$9 \times 5 = 45$ $\frac{11}{2} + \frac{45}{180} + \frac{40}{2}$ $\frac{91}{2}$

7. What is the output of the following code? Use the space on the right to trace the code. You must show your work in the trace to get full credit. (10pts)

```
def fun1(a, c, b):
    q = a - b + 4
    y = q//2 + a*c
    return y


def fun2(x, a, y):
    if x >= y:
        x = a - 2
    else:
        x = a - 4
    return x


def main():
    b = fun1(9, 5, 2)
    x = fun2(b, 9, b)
    print("The answer is ", x)
    print("or maybe its ", b)
```

a - 2 + 4
7 + 4
11

a: 9 — c: 5 — b: 2 — q: 11 — y: 11 + 91/2

91/2

x: 91/2 — a: 9 — y: 91/2

x = 91/2 - 18/2 = 73/2

91/2 - 18 = 73

x: 91/2 — 87/2

b: 91/2 — x: 87/2

Display the output here:

The answer is 87/2

or maybe its 91/2

8. Given the following function definition: (10pts)

```
def mysteryFunction(inStr):
    new = ''
    i = len(inStr)-1
    while i >= 0:
        c = inStr[i]
        new = new + c
        i = i - 1
    return new
```

'bill'
new = ''
i = 3 2 1 0
lli b
-1

'bob'
new = ' '
i = 2 1 0
c = b o b
new = bob

a) What are the results of the following function calls:

`X1 = mysteryFunction('bill')`	X1: lli b
`X2 = mysteryFunction('bob')`	X2: bob
`X3 = mysteryFunction('1234')`	X3: 4321

b) Explain what this function does in general when given a string? Use the space below to trace the function for each function call.

'bill'

new	i	c
	3	
	2	
	1	
	0	
lli b	-1	b

'bob'

new	i	c
	2	
	1	
	0	
bob		b

'1234'

new	i	c
	3	
	2	
	1	
	0	
4321		

This function returns the given string backwards

9. The following function counts how many times a particular character appears in a string.

```
def countChar(inStr, ch):
    count = 0
    for i in range(len(inStr)):
        if inStr[i] == ch:
            count = count + 1
    return count
```

So, a call to this function:

```
countChar('abracadabra','a')
```

returns 5.

Create a new function, `countSub`, by modifying the `countChar` function so that it counts the number of times a given character appears in a substring, delineated by the position of the start of the substring, and the position of the end of the substring. For example, the call to your function:

```
countSub('abracadabra','a',1,5)
```

would return 2 because there are two `'a'`s between position 1 and position 5 in the string. You can fill in the following skeleton:

```
def countSub(inStr, ch, start, end):
    count = 0
    if len(inStr) < range (end)
        return ('Substring doesn't exist')
    else:
        for i in range(
        start = star
        while start < end:
            if inStr[start] == ch:
                count = count + 1
            start = start + 1


    return count
```

(10 pts)

10. Using the `countChar` function above, write another function, called `countAllChars` that is given a two strings, `str1` and `str2` and returns a list containing the number of times each of the characters in `str1` appears in `str2`. For example, the following function call:

```
countAllChars('cat', 'abracadabra')
```

would return the following list:

```
[1, 5, 0]
```

Because there is 1 'c' in 'abracadabra', there are 5 'a's, and there are no 't's.

Note: You should not copy the code from the function in problem 10. You should call that function to do the work in this function. (10 pts)

```
def countAllChars(str1, str2):
    countList = []
    for i in range(len(str1)):
        x = countChar(str2, str1[i])
        countList.append(x)

    return countList
```

11. Assume you are working for an online toy store. Your company tracks the inventory for each toy in the warehouse. Your job is to find all toys that are low in inventory, and also the total number of toys currently in the warehouse. You can assume that the data file looks like the following:

```
Dollhouse, 45
Firetruck, 33
Teddy Bear, 66
Fun Blocks, 83
YoYo, 99
...
```

Below is some skeleton code for the functions of this program. Write the code for **TWO** of the functions in the program. You can choose which functions to fill in. (10pts)

```
# Function to get the data from the file given the name of the file
# Parameters: Name of data file
# Return: List of toy names, list of number in inventory
def getData(fname):
   return toyList, invList

# Function to find the all toys that are low in inventory
# Parameter: List of all inventory values for the toys, threshold for low
#            inventory
# Return: List of indexes representing toys with low inventory
def findLowInventory(invList, threshold):
   return lowInvList

# Function to compute the total number of books sold
def countAllToys(invList):
   return totalToys
```

(Write your first function here)

```
def getData(fname):
    myFile = open(fname, 'r')
    toyList = []
    invList = []
    for line in myFile:
        line = line.strip()
        toy, inv = line.split(',')
        toyList.append(toy)
        invList.append(int(inv))
    myFile.close()
    return toyList, invList
```

(Write your second function here)

```
def findLowInventory(invList, threshold):
    lowInvList = []
    for i in range(len(invList)):
        if invList[i] < threshold:
            lowInvList.append(i)
    return lowInvList
```

12. Write the `main` function for the toy store program from problem 11 above. You may hardcode the name of the data file, `'etoys.csv'` in the main function so that no exception handling is necessary. Note that the main function is responsible for asking the user to enter the low inventory threshold. Assume that there is also a `printResults` function defined as follows:

```
# Function to print the results
def printResults(lowInvList, toyList, totalToys)
   return
```

(10pts)

```
def main():
    fname = etoys.csv
    # toyList, invList = getData(fname)
    threshold = int(input('what is the threshold? '))
    lowInvList = findLowInventory(invList, threshold)
    totalToys = countAllToys(invList)
    printResults(lowInvList, toyList, totalToys)
```