Name: ***(as it would appear on official course roster)***
Umail address: **@umail.ucsb.edu**
Optional: name you wish to be called if different from name above.
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"

1

h03

CS8 M18

h03: Perkovic 4.3-4.4 (Files, Errors, Exceptions), 5.1-5.2 (decision control, accumulators, nested loops), 5.3-5.6 (2-d lists, more on loops)

ready?	assigned	due	points
true	Thu 08/16 09:30AM	Tue 08/28 09:30AM	285

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE.
There is NO MAKEUP for missed assignments, and you may not submit work in advance, or on behalf of another person.
In place of that, we drop the lowest score (if you have a zero, that is the lowest score.)

READING ASSIGNMENT

Please read Perkovic 4.3-4.4 (Files, Errors, Exceptions). Then complete these problems.

1. (10 pts) Please fill in the information at the top of this homework sheet, as usual. WRITE DARK, and write your name at the top of all sheets and turn in all sheets UNCONNECTED. No staples, paper clips, fold/tear etc or anything that would jam up the scanner.

2. One important abstraction provided by most operating systems (Windows, Linux, or MacOS, for example) is something called a “file system”.

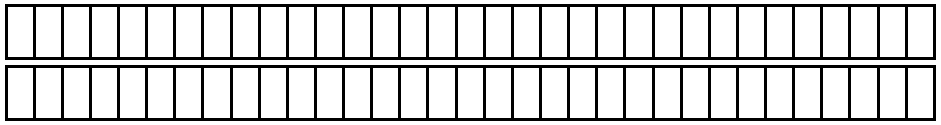
    a. (10 pts) As was discussed in Chapter 1, one of the purposes of an abstraction is to provide a uniform way of treating something that hides irrelevant detail. Our text points out an important detail of storing information that a file system helps us to ignore. What is that detail?

    b. (10 pts) According to our text, every file on a file system can be referred to be an “absolute pathname”, which consists of a sequence of … what?

    c. (10 pts) In contrast to an “absolute pathname”, we have the concept of a “relative pathname”. What is the technical term used for the “starting point” of a “relative pathname”?

3. (10 pts) Section 4.2 discusses formatted output. What is the output of the following print statement? Please put one character per box to show the exact spacing. Try to figure it out by hand before checking your answer online. If you spoil the first grid, use the second.

```
print('{0:4},{2:6}'.format(123,456,789))
```



2

h03

CS8 M18

4. Pages 112-114 discuss four ways of reading text from a file.

3

h03

CS8 M18

   a. (10 pts) The first way is shown in the listing at the bottom of p. 112. On line 4 of that listing we see:

```
content = infile.read()
```

   After this line of code is executed, what would `type(content)` return? (i.e. would it be `<class 'int'>`, `<class 'float'>`, `<class 'str'>`, `<class 'list'>`, or something else?)

   b. (10 pts) The second way is shown in the middle of p. 113. On line 7 of that listing we see:

```
wordList = content.split()
```

   What does the `.split` method do, and what is stored in `wordList` as a result?

   c. (10 pts) The third way is shown in the listing near the top of p. 114. On line 4 of that listing we see:

```
lineList = infile.readlines()
```

   After this line of code is executed, what would `type(lineList)` return? (i.e. would it be `<class 'int'>`, `<class 'float'>`, `<class 'str'>`, `<class 'list'>`, or something else?)

   d. (10 pts) The fourth and final way is shown in an interactive example on the lower half of p. 114, and looks like this (with the Python prompts removed):

```
infile = open('example.txt')
for line in infile:
   print(line,end='')
```

   The book suggests that this fourth method has an advantage over the other three in a particular circumstance. What is the circumstance in which we would want to use this method instead of one of the other three?

5. (10 pts) Section 4.4 discusses two types of problems that can arise in a Python program: *errors* and *exceptions*. One difference is that in Python an *error* just results in a message bring printed on the screen, while an *exception* also results in an object being created that stores information about the exception. Another difference is the root cause, or context in which errors happen vs. exceptions. Describe the difference between what kind of problem causes an *error*, vs what kind of problem causes an *exception*.

READING ASSIGNMENT Please read Perkovic 5.1-5.2 (decision control, accumulators, nested loops). Then complete these problems.

4

h03

CS8 M18

6. (40 pts) p. 129 shows a function definition for a multi-way if/else that prints a message depending on the temperature.

   Rewrite this function so that instead of printing a message, it returns a letter grade (e.g. `return 'A'` instead of `print('It is hot')` based on the integer parameter. If the grade is 90 or above, return an 'A'. If it is 60 or higher, but less than 90, return a 'C', and if it is less than 60, return an 'F'. (In real life, there would be Bs and Ds, but this is just an exercise.)

   NOTE: Be careful about the fact that in an `if`/`elif`/`else`, some of the relationships are implicit. You cannot get to the `elif` unless the condition on the first `if` is false. So you should not check for that a second time. (To be more clear: the `elif` on p. 129 says: `elif t > 32:` rather than `if t <= 86 and t > 32`. The `t<=86` part is unnecessary, because we would never even get to the `elif` unless `t<=86` were true. Make sure you keep this in mind as you write your code for this problem. Points may be deducted if you do redundant checks, even if the code "works".)

7. For the Python code in the left box, write the output in the right box

(10 pts)
`colors = ["red","green","blue"]`
`for c in colors:`
`  print(c)`	

(10 pts)
`fruits = ["apple","banana","pear","grape"]`
`for i in range(4):`
`  print(i,fruits[i],sep=",")`	

5
h03
CS8 M18

8. (10 pts) p. 134-136 discusses the "Accumulator Pattern", which is a very important topic in this course; one of the most important for you to master. So please read those two pages several times and try to understand every detail. The figure at the top of p. 135 shows the various stages of execution for the code on p. 134.

   The code `mySum = mySum + num` takes the old value of `mySum`, adds `num` to it, and stores the result back in `mySum`.

   That code is done inside a `for` loop, `for num in myList`, after setting `mySum` initially to zero.

   The final value for `mySum` is `20`. What does that number `20` represent in this case?

9. (10 pts) On p. 135, we see the intermediate values for mySum, namely `3`, `5`, `12, 11,` and `10`. Try to understand where those values come from as the loop progresses.

   Then, imagine the same loop were executed, but with the first line of code being `numList=[9, 3, 1, 1, 7]` (instead of `numList=[3, 2, 7, -1, 9]`.) What would the successive intermediate values of `mySum` be in that case? List them in the space below.

10. (10 pts) On page 135-136, the textbook discusses accumulating a product instead of a sum. The accumulator variable is called `myProd` this time. In the version of the code that works properly, what is `myProd` initialized to, and why?

READING ASSIGNMENT Please read Perkovic 5.3-5.6 (2-d lists, more on loops). Then complete these problems.

6
h03
CS8 M18

11. Each of the problems below shows a function definition at left, then one or more function calls at right. For each function call, write what the function call evaluates to.

a.

```
def foo(x):
    color = "blue"
    if x > 10:
        color = "red"
    color = "green"
    return color
```

pts	function call	write result here
(5 pts)	`foo(5)`	
(5 pts)	`foo(10)`	
(5 pts)	`foo(15)`	

b.

```
def bar(x):
    color = "blue"
    if x > 10:
        color = "red"
    else:
        color = "green"
    return color
```

pts	function call	write result here
(5 pts)	`bar(5)`	
(5 pts)	`bar(10)`	
(5 pts)	`bar(15)`	

c.

```
def baz(x):
    count = 0
    i = 1
    while i < x:
        if i % 5 == 0:
            count += 1
        i+=1
    return count
```

pts	function call	write result here
(5 pts)	`baz(3)`	
(5 pts)	`baz(10)`	
(5 pts)	`baz(12)`	

12. (10 pts) Section 5.3 describes how a "list of lists" can be used to represent a 2-dimensional list, such as a matrix or grid.

Suppose we wanted to program a Python program to play Tic-Tac-Toe. Tic-Tac-Toe is played in a grid similar to the ones shown in the figures below. Figure (a) shows an empty Tic-Tac-Toe grid, while Figures (b),(c),(d) show what might be the first three moves of the game. Players take turns filling in squares with 'x' and 'o'.

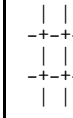
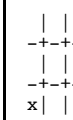
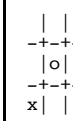
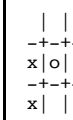
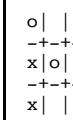
We can represent the Tic-Tac-Toe boards shown here as lists of lists of strings. A list of list of strings representation is shown below for three of the five boards. The top level list represents a list of three rows. Each of the lists has three elements, each of which is a space, `x` or `o` for that square.

The tables below shows the representations for the boards in Figures (1),(3) and (5). Fill in the Python representations for Figures (2) and (4).

7

h03

CS8 M18

Figure 1		`board = [[' ',' ',' '],[' ',' ',' '],[' ',' ',' ']]`
Figure 2		(10 pts) `board =`
Figure 3		`board = [[' ',' ',' '],[' ','o',' '],['x',' ',' ']]`
Figure 4		(10 pts) `board =`
Figure 5		`board = [['o',' ',' '],['x','o',' '],['x',' ',' ']]`

13. (10 pts) Since lists are mutable, the change from Figure 2 to Figure 3 could be made by an assignment statement. Which of the following assignments statements would do the job? (Circle one)

`board[1][1]='o'` `board[1,1]='o'` `board[2][2]='o'` `board[2,2]='o'` none of these

14. (10 pts) Likewise, which assignment statement changes the value of `board` from the one shown in Figure 3 to the one shown in Figure 4? (Circle one)

`board[0][1]='x'` `board[0,1]='x'` `board[1][0]='x'` `board[2,1]='x'` none of these