

CMPS 8 W18
Introduction to Computer Science

Final Examination

Please state your answers as clearly as possible. This exam not only tests your understanding of the material, but also how well you can convey your understanding to us. Remember that you are solely responsible for the answers to the questions, therefore, please refrain from consulting with your class peers.

Please write all your answers **LEGIBLY** and **CLEARLY**. If we cannot decipher your answers, you will not receive credit.

No electronic devices are allowed during the exam (calculators, cell phones, laptops, etc.).

READ all questions carefully before attempting to answer. If there are any ambiguities in the statement of questions, please ask us. **You may assume that each problem is correct and solvable unless the question specifically asks about errors.**

THE GRADE IN THIS EXAM IS A TOTAL OF 94 POINTS.

Name (as it would appear on the official course roster)	Umail Address
	@umail.ucsb.edu

Question 1 (6 points)

a. Complete the definition of the function below, consistent with its docstring comment, by filling in each blank with **exactly one variable, operator, or constant value**.

```
def contains_exclamations(s):  
    ''' Returns True if the string s contains any exclamation marks '!'. Returns  
        False otherwise.  
    ...  
    return _____.find(_____) _____ 0
```

b. Complete the definition of the function below, consistent with its docstring comment, by filling in each blank with **exactly one variable, operator, or constant value**.

```
def yell(s):  
    ''' Returns a string where all all '.' are replaced with '!' in the parameter  
        string s.  
    ...  
    return _____.replace(_____, _____)
```

Question 2 (15 points)

```

a = 'CS8,CS16,CS24'
b = [2,4,6,8,10,12]
c = (1,3,5,7)
d = {"A", "B", "C", "D", "E", "F"}
e = a.split(',')
f = {e[0]: {"A", "B", "C"},
     e[1]: {"D", "E", "F"},
     e[2]: {"G", "H", "I"} }

```

For each of the Python expressions in this table, write the value the expression evaluates to as if it were executed in IDLE's command line window and its data type. Choose from **int**, **float**, **bool**, **str**, **function**, **list**, **dict**, **set**, or **tuple**. Treat each expression below independently of the preceding expression(s).

Expression	Value	Type (From list above)
<code>b[1] / 2 + 1</code>		
<code>len(c)</code>		
<code>len(f)</code>		
<code>len(e)</code>		
<code>b[2:5]</code>		
<code>e</code>		
<code>a[-5]</code>		
<code>a.find(",")</code>		
<code>f["CS24"]</code>		
<code>d >= f["CS16"]</code>		
<code>d < f["CS16"]</code>		
<code>"E" in f["CS8"]</code>		
<code>c[len(b) - 3]</code>		
<code>d - {"A", "C", "E"}</code>		
<code>{"A", "C", "E", "G"} - d</code>		

Question 3 (10 points)

Match the correct output for each code segment using the following choices.

Assume `example.txt` consists of three lines and exists in your working directory and the `infile` variable is defined below. Assume there is a newline character (`'\n'`) immediately after each line of text except the last line. Treat each expression below independently of any preceding expression(s). Write your answer given the choices below [A – D] in the blank space next to each block of code. A choice [A – D] may be used multiple times.

example.txt: (this line is not included in example.txt)

Line 1
Line 2
Line 3

```
infile = open('example.txt', 'r')
```

a. `print(infile.read(10))` _____

b. `print(infile.read())` _____

c. `print(infile.readline())` _____

d. `for x in infile.readlines():`
 `print(x)` _____

e. `for x in infile:`
 `print(x)` _____

A.
Line 1

Line 2

Line 3

C.
Line 1

B.
Line 1
Lin

D.
Line 1
Line 2
Line 3

Question 4 (6 points)

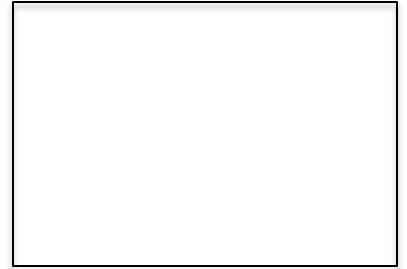
a. Briefly state (in at most two sentences) what data PERSISTENCE means.

b. Briefly describe (in at most two sentences) the effect of the 'a' and 'w' parameters in the `open` function above instead of 'r'.

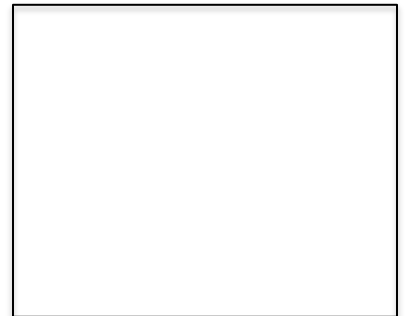
Question 5 (15 points)

Write the output for the following segments of code in the boxes to the right.

a.
counter = 0
for i in range(2,4):
 for j in range(2):
 counter += i + j
 print(counter)



b.
L = [1,2,3,4,5,6,7,8]
while True:
 if len(L) == 1:
 break
 x = L.pop()
 if x % 4 == 0:
 continue
 if x % 3 == 0:
 print(len(L))
 elif x % 2 == 0:
 print(L[len(L) - 1])



c.
L = [1,2,3,4]
result = []
for i in L:
 if i % 2 == 1:
 result.append(i * 2)
 print(result)
 else:
 result.append(i / 2)
 print(result)



d.
def f(n):
 if n <= 0:
 return 0
 print(n)
 f(n-2)
 print(n)

f(4) # assume this is called in a .py file



Question 6 (8 points)

Complete the following function definition according to the docstring comment and assert examples below:

```
def recursiveCountCharacters(s, c):  
    ''' The parameter s represents an arbitrary string and the parameter c  
        represents a single character. Return the number of occurrences c is in s.  
        Note: your solution must use recursion in order to receive credit '''
```

```
assert recursiveCountCharacters("hello", 'l') == 2  
assert recursiveCountCharacters("apple", 'a') == 1  
assert recursiveCountCharacters("", 'a') == 0
```

Question 7 (8 points)

Suppose we have a list of names, some of which may occur more than once on the list. For example:

```
NL = ['Joe', 'Sam', 'Joe', 'Jill', 'Joe', 'Joe', 'Jill', 'Sam', 'Jane', 'Jane', 'Jane', 'Joe', 'John']
```

And suppose that we want to know which name occurs most frequently. We can create a dictionary that gives us a collection of each distinct name on the list, along with the number of times it occurs. Complete the following function definition according to the docstring comment and assert example below:

```
def tally_names(L):  
    ''' The parameter L is a list of strings. Return a dictionary with each unique  
        string in L as the key and the number of times that string occurs in L as the  
        value. '''
```

```
assert tally_names(NL) == {'Sam': 2, 'Jill': 2, 'Joe': 5, 'Jane': 3, 'John': 1}
```

Question 8 (10 points)

Assume we would like to write a function that prints out market information in a table as follows:

```
Albertsons:
    Item: milk   | ID: 1 | Price: $3.50
    Item: salt   | ID: 4 | Price: $3.00
    Item: sugar  | ID: 5 | Price: $2.50
-----
Ralphs:
    Item: bread  | ID: 2 | Price: $1.00
    Item: cola   | ID: 3 | Price: $5.00
    Item: salt   | ID: 4 | Price: $3.00
-----
Pavilions:
    Item: lettuce | ID: 6 | Price: $0.50
    Item: sugar   | ID: 5 | Price: $2.50
-----
```

The following code prints out the markets and their inventory in the format above:

```
Item = namedtuple('Item', 'name id price')
Market = namedtuple('Market', 'name inventory')

milk = Item('milk', 1, 3.5)
bread = Item('bread', 2, 1.0)
cola = Item('cola', 3, 5.0)
salt = Item('salt', 4, 3.0)
sugar = Item('sugar', 5, 2.50)
lettuce = Item('lettuce', 6, 0.50)
albertsons = Market('Albertsons', [milk, salt, sugar])
ralphs = Market('Ralphs', [bread, cola, salt])
pavilions = Market('Pavilions', [lettuce, sugar])
markets = [albertsons, ralphs, pavilions]

for m in markets:
    print("{}:".format(m.name))
    for i in m.inventory:
        print(FORMAT_STRING.format(i.name, i.id, i.price))
    print("-" * 10)
```

Circle the following values for `FORMAT_STRING` that will produce the printed table above (**one or more answers may be correct**).

- A. `"\tItem: {:8s}| ID: {:3d}| Price: ${:2.3f}"`
- B. `"\tItem: {:8s}| ID: {:<3d}| Price: ${:2.2f}"`
- C. `"\tItem: {:<8s}| ID: {:<3d}| Price: ${:2.2f}"`
- D. `"\tItem: {:>8s}| ID: {:>3d}| Price: ${:2.2f}"`
- E. `"\tItem: {:8s}| ID: {:<3d}| Price: ${:3.2f}"`

Question 9 (8 points)

Given the following function definitions below, complete each assert statement such that they will pass.

a.

```
def f(L):  
    ''' The parameter L represents a list of strings '''  
    x = ""  
    for w in L:  
        if len(w) >= len(x):  
            x = w  
    return x
```

assert f(["this", "is", "a", "sentence"]) == _____

assert f(["hey", "hee", "haa"]) == _____

assert f([]) == _____

assert f(["a", "aa", "b"]) == _____

b.

```
def g(L):  
    ''' The parameter L represents a list of strings '''  
    y = ""  
    for s in L:  
        for c in s:  
            if c not in y:  
                y += c  
    return y
```

assert g(["this", "is", "a", "sentence"]) == _____

assert g(["hey", "hee", "haa"]) == _____

assert g([]) == _____

assert g(["a", "aa", "b"]) == _____

Question 10 (8 points)

An animal shelter is trying to keep track of the animals under their care and has asked you to design a program to maintain animals' information. For each animal, you will store a unique `int` that identifies a specific animal id, the animal's name (`str`), the species of the animal (`str`), and the age of the animal (`int`).

Each animal is a `namedtuple` defined as follows:

```
Animal = namedtuple('Animal', 'id name species age')
```

The animals are stored in a database represented as a List of animal `namedtuples`. Fill in the blanks in the code below to implement the functions correctly. Each blank must be filled with **exactly one Python variable, operator, or constant**.

```
def create_database():
    ''' Return an empty database (i.e. an empty list) '''
    return []

def add_animal(animal, L):
    ''' The parameter animal is an animal namedtuple to be added to the database. The parameter
    L is a list of animals representing the animal database. Return the database (list) with
    the new Animal added at the end.'''

    return L + [_____]

def remove_animal(animal, L):
    ''' The parameter animal is an animal namedtuple to be removed from the database. The
    parameter L is a list of animals representing the animal database. Return the database
    (list) without the animal'''
    DB = []
    for a in _____:
        if a.id != _____ .id:
            DB.append(a)
    return _____

def find_animal_by_name(name, L):
    ''' The parameter name is a name of an animal to find. The parameter L is a list of animals
    representing the animal database. Return a list of Animals that have this name'''
    DB = []
    for a in L:
        if a.name _____ name:
            DB.append(a)
    return DB

def find_animal_by_id(animal_id, L):
    ''' The parameter animal_id is the id of an animal to find. The parameter L is a list of
    animals representing the animal database. Return the Animal namedtuple with this ID'''

    for a in L:
        if a.id _____ animal_id:
            return _____
    return None

def is_database_empty(L):
    ''' The parameter L represents the animal database. Return true if the database (list) is
    empty '''
    return _____(L) == 0
```