CMPSC 8 W18
Introduction to Computer Science

Midterm 2 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 57 POINTS.

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

Question 1 (10 points)

Use the following variable definitions to answer this question:

```
s = 'AP-CS,CS8,CS16'
L = [2, 4, 6, 8, 10, 12, 14, 16]
```

Provide the **EXACT** output for the following statements when executed in IDLE's Interactive Shell. Write your answer in the box to the right of the statements.

`s.replace(',', '**')`	
`s[3:7].lower()`	
`s.strip('CS')`	
`s.split('CS')`	
`s.find('CS')`	
`s.find(',')`	
`s.find('24')`	
`s[L[len(s[2:5])]]`	
`s.count('C')`	
`s.count(',C')`	

Question 2 (8 points)
A student's status at UCSB is calculated based on the number of completed units. The undergraduate student standing breakdown is as follows:

Freshman: 0 – 44.9 units
Sophomore: 45 – 89.9 units
Junior: 90 – 134.9 units
Senior: 135+ units

Below are four versions of a function that takes a student's number of units and returns the name of the corresponding status. In the table below, mark **P** or **F** in each cell indicating which assertions pass (**P**) or which assertions fail (**F**) for each version of the function.

Assertions	A	B	C	D
`assert student_standing(136) == 'Senior'`				
`assert student_standing(135) == 'Senior'`				
`assert student_standing(134) == 'Junior'`				
`assert student_standing(90) == 'Junior'`				
`assert student_standing(89) == 'Sophomore'`				
`assert student_standing(45) == 'Sophomore'`				
`assert student_standing(44) == 'Freshman'`				
`assert student_standing(0) == 'Freshman'`				

A.
```
def student_standing(units):
    '''Return the student standing with
       respect to the number of units '''
    if units <= 44.9:
        return 'Freshman'
    if units <= 89.9:
        return 'Sophomore'
    if units <= 134.9:
        return 'Junior'
    return 'Senior'
```

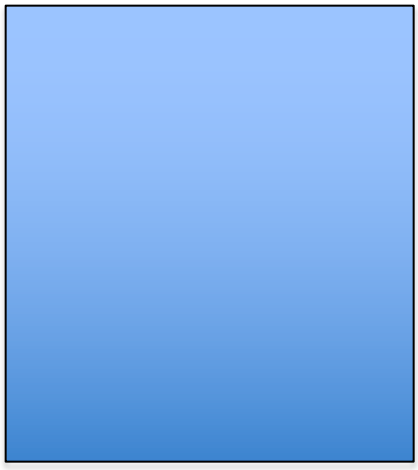
B.
```
def student_standing(units):
    '''Return the student standing with
       respect to the number of units '''
    if units <= 44.9:
        return 'Freshman'
    elif units <= 89.9:
        return 'Sophomore'
    elif units <= 134.9:
        return 'Junior'
    else:
        return 'Senior'
```

C.
```
def student_standing(units):
    '''Return the student standing with
       respect to the number of units '''
    if units > 0 and units < 45:
        return 'Freshman'
    if units > 45 and units < 90:
        return 'Sophomore'
    if units > 90 and units < 135:
        return 'Junior'
    else:
        return 'Senior'
```

D.
```
def student_standing(units):
    '''Return the student standing with
       respect to the number of units '''
    standing = ''
    if units >= 135:
        standing = 'Senior'
    if units >= 90 and units <= 135:
        standing = 'Junior'
    elif units >= 45 and units <= 90:
        standing = 'Sophomore'
    elif units >= 0 and units <= 45:
        standing = 'Freshman'
    return standing
```

Question 3 (8 points)
What does the following code print out? Write the output in the blank box to the right of the code.

```
counter = 0
while True:
    counter += 2
    if counter % 5 == 0:
        break
    else:
        print(counter)
    if counter % 3 == 0:
        continue
    print(counter)
print("done")
```



Question 4 (8 points)
Write the correct output for the following print statements. Put one character per box to show the **exact** spacing.

a. `print("Founded in{:5d}.".format(1999))`

b. `print("UC{:4s} (${:2.1f})".format("SB",4321.142))`

c. `print("UC{:>4s} (CS{:1})".format("SB",190))`

d. `print("{1},-,{0},-,{2}.".format(999,111,555))`

Question 5 (6 points)
Write out all possible values for the following statements. You can assume the random module is already imported in your code.

a. `random.randrange(2,5) + 10`

b. `random.randrange(0,4) * random.randrange(0,3)`

c. `random.randrange(0,4) + random.randrange(1,5)`

Question 6 (9 points)
Recall the function definition of creating a 2D list representing a screen of pixels:

```
def create_screen(rows, columns):
    ''' Creates a screen of rows x columns pixels '''
    grid = []
    for x in range(rows):
        grid.append([0] * columns)
    return grid
```

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

```
def count_pixels(screen):
    ''' The parameter screen has arbitrary values and dimension (represented as a
    2D list). Return the number of pixels with a value equal to 0. '''
```

```
assert countPixels([[0,0],[0,1],[0,0]]) == 5
assert countPixels([[1,0,0],[0,1,0],[0,0,1]]) == 6
```

Question 7 (8 points)
Complete the following function definition according to the docstring comment and assert examples below:

```
def oddStrings(L):
    ''' The parameter L represents a list of strings. Return a list of strings
    that contains only the strings in L with an odd number of characters '''
```

```
assert oddStrings(["1"]) == ["1"]
assert oddStrings(["1","12"]) == ["1"]
assert oddStrings(["1","12","123"]) == ["1","123"]
```