

# Computer Science 1 — CSci 1100 — Fall 2022

## Exam 1

September 29, 2022

Name: Anannya Punia

RCS ID: 

p	u	n	i	a	a		
---	---	---	---	---	---	--	--

 @rpi.edu

RIN#: 662041944

Honor pledge: On my honor I have neither given nor received aid on this exam.

Please sign here to indicate that you agree with the honor pledge: Anannya Punia

### Instructions:

- You have 90 minutes to complete this test.
- Clearly print your name, RCS ID (in all caps) and your RIN at the top of your exam.
- You must **have your Student ID**, you must be seated in the **correct section**, and you must put away all calculators, phones, etc. and take off/out all headphones and earbuds. Failure may incur a **20 point penalty for each infraction**. If you are unsure if you are in the right section, please see a proctor before the exam starts.
- You may use only one double-sided crib sheet. Put your name and your rcs id on it and turn it in at the end of the exam. Otherwise, put away all books, laptop computers, and electronic devices.
- Please read each question carefully several times before beginning to work.
- In all the questions, your output must match exactly the given output (do not insert additional spaces if they are not in the output).
- We generally will not answer questions except when there is a glaring mistake or ambiguity in the statement of a question.
- Except for problem 1, there are no Python syntax errors anywhere on this test.
- Unless otherwise stated, you may use any valid Python technique to solve any problem.
- Please state clearly any assumptions that you have to make in interpreting a question.
- There are **7 questions** on this test worth a total of **100 points**.
- When you are finished with this test please turn it into one of the proctors along with your crib sheet. After you show the proctor your student id you will be free to leave the exam room.

1. (10 points total; 2 points each) Assume each of the following is a complete, separate program. For each program find the first error that prevents the program from generating output, or decide there is no error. If there is an error, write in the solution box that there is an error, write the line number and very briefly describe the error. If the program would run and generate output, write **No Error**.

<b>Part a:</b>  <pre>s=('abc','def') #1 x,y_z,t = s      #2 print(s[1])      #3 print([y_z])     #4</pre>	<b>Answer:</b>  #2 t cannot be assigned to anything
<b>Part b:</b>  <pre>s1='hello "python" ' #1 s2='Is fun'          #2 new_s=s1+s2          #3 s3=s1 s2             #4 print(s1*3)          #5</pre>	<b>Answer:</b>  #4 will not recognize S1 and S2
<b>Part c:</b>  <pre>s1='Good\tmorning\n' #1 s2='Have\ta\nnice\nday' #2 s3=('\\\\\\n'*3)      #3 s4= s1*s3            #4 print(s4)            #5 print(s1+s2)         #6</pre>	<b>Answer:</b>  #3 will not recognize '\\\\n' because there is an extra '\'
<b>Part d:</b>  <pre>x = 9                #1 y = 9 / 3            #2 a = "abab"           #3 b = int(y) * a       #4 print(b)             #5</pre>	<b>Answer:</b>  no error
<b>Part e:</b>  <pre>from math import *   #1 import math          #2 def new(s):          #3     math.pi=3        #4     return sqrt(s) *math.pi #5 print(new(math.pi))  #6</pre>	<b>Answer:</b>  no error

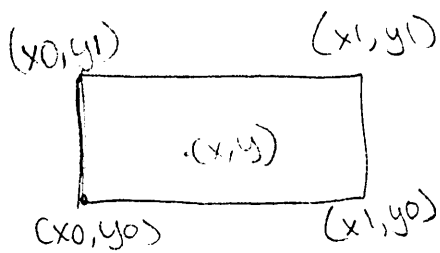
2. (15 points total; 3 points each) Assume each of the following is an individual program. Write the exact output of these programs in the area provided. Use the scratch area as you wish. There are no syntax errors in the code provided in this question.

Part a:	Answer:	Scratch area:
<pre>print('*\n'*3,end='*') print('-', '+', '\\', sep='*')</pre>	<pre>* * * - * + * \\\</pre>	
Part b:	Answer:	Scratch area:
<pre>def t(x):     n=int(x[0]) = 25     m=x[1] 0     return n*m, str(n)+m     0, 25 print(t(('2', '2'))) print(t('25'))</pre>	<pre>4, 24 0, 25</pre>	
Part c:	Answer:	Scratch area:
<pre>def _find(s):     x=s.count('*') =     y=s*x     return y.replace('*', '*'+x)  s='1*2*' print(_find(s))  t='12*31' print(_find(t))</pre>	<pre>1**2**1**2** 12***31</pre>	<pre>S = '1*2*' x = 2 y = 1*2*1*2*  S = 12*31 x = 1 y = 12*31 12***31</pre>
Part d:	Answer:	Scratch area:
<pre>a='apple' b='Cat' c='Zoo'  print(a&lt;b or b&lt;c) print(a&gt;c and a&lt;b) print(a==c or a!=b)</pre>	<pre>True False True</pre>	<pre>ap &lt; Cat C &lt; Z apple &gt; Z</pre>

Part e:	Answer:	Scratch area:
<pre> import math as m x='helloworld' x1=x.count('o') x2=m.sqrt(x.find('o')) x.upper() x3=x.count('0') print(x*x1) print(x2) print(x3) </pre>	<p>HELLOWORLDHELLOWORLD</p> <p>2</p> <p>2</p>	<p>x1 = 2</p> <p>x2 = 2</p> <p>HELLO WORLD</p> <p>x3 = 2</p>

3. (20 points total; 4 points each) Give a single line of code to accomplish each of the following. You can assume you are typing into the Python interpreter. Make sure your solution is only one line. Answers of more than one line will result in a loss of credit. Do not use **ifs** or **loops**.

<b>Part a:</b>	<b>Answer:</b>
<p>Given a tuple variable <code>t</code> with exactly 2 string type values, write a single line of code to print average number of letter <code>a</code>'s in the tuple. For example, with the values <code>t=('aaAAaeroplane','aheAd')</code>, your expression would print:</p> <p>4.0</p> <p>Here there are 6 <code>a</code>'s in the first string and 2 <code>a</code>'s in the second. Therefore the average per word is 4.0</p>	<pre>print((t[0].count('a') + t[1].count('a')) / 2)</pre>
<b>Part b:</b>	<b>Answer:</b>
<p>Given a string <code>s</code> of of 4 digits, write a single line of code to output the larger of either the first 2 digits or the last 2 digits. For example if <code>s = '1234'</code>, the output should be the integer (not string) 34. If <code>s = '9934'</code>, the output should be the integer (not string) 99.</p>	<pre>print(max(s.split()[0], s.split()[1]))</pre>
<b>Part c:</b>	<b>Answer:</b>
<p>Given a string <code>a</code>, write an expression that prints the string repeated on three consecutive lines in all capitals. Note that this will require a <code>print()</code>. For <code>a='hello world, this is python'</code> this would be:</p> <pre>HELLO WORLD, THIS IS PYTHON HELLO WORLD, THIS IS PYTHON HELLO WORLD, THIS IS PYTHON</pre>	<pre>print(a.upper() * 3 sep = '\n')</pre>

<p><b>Part d:</b></p> <p>Given a point, <math>(x, y)</math> and a rectangle with corner coordinates <math>(x_0, y_0)</math>, <math>(x_0, y_1)</math>, <math>(x_1, y_0)</math> and <math>(x_1, y_1)</math> where <math>x_0 &lt; x_1</math> and <math>y_0 &lt; y_1</math>, write a boolean expression using the variables <math>x</math>, <math>y</math>, <math>x_0</math>, <math>x_1</math>, <math>y_0</math>, <math>y_1</math> that evaluates to True if the point is strictly within the rectangle. Otherwise, it evaluates to False.</p> 	<p><b>Answer:</b></p> <p><math display="block">if x_0 \leq x \leq x_1 \text{ and } y_0 \leq y \leq y_1</math></p>
<p><b>Part e:</b></p> <p>Given a string <math>a</math>, write an expression that evaluates to the string with all spaces, tabs, and back slashes replaced with commas, and with the initial character of the string in upper case. For example, if:</p> <pre>a = "what is your quest?"</pre> <p>For <math>a</math>, this would give:</p> <pre>'What,is,your,quest?'</pre>	<p><b>Answer:</b></p> <p><math display="block">print(a.title() \text{ sep} = ',')</math></p>

This page is left blank for scratch work. Do not put your solutions here. Put them in the boxes provided for each question.

4. (16 points) You visit the supermarket to buy groceries. In this problem you will write code to calculate the change the store should give back depending on the amount due. Assume, you can only pay in one dollar bills. Your change can be returned only in one dollar bills (100 cents), quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent) (no other denominations). Assume your code takes the amount due and amount paid (by you) both as input variables.

Depending on the amount due and the money you paid, you should print appropriate output messages.

You can use if constructs including elif and else clauses, but do not use loops.

Here are some example runs :

```
>>>Amount due=>5
```

```
>>>Dollar bills paid=>5
```

```
No change due
```

```
>>>Amount due=>5.75
```

```
>>>Dollar bills paid=>7
```

```
Collect 1 one dollar bills
```

```
Collect 1 quarters
```

```
Collect 0 dimes
```

```
Collect 0 nickels
```

```
Collect 0 pennies
```

```
>>>Amount due=>5
```

```
>>>Dollar bills paid=>3
```

```
Paid 2.0 dollars less than amount due
```

```
>>>Amount due=>8.82
```

```
>>>Dollar bills paid=>11
```

```
Collect 2 one dollar bills
```

```
Collect 0 quarters
```

```
Collect 1 dimes
```

```
Collect 1 nickels
```

```
Collect 3 pennies
```

*if dollar bills paid > amount due  
change = amount due - dollar bills  
change % 1 = dollar bills  
if change < 25  
change - 25 = nchange*

*82  
- 25*

Write your answer on the next page.



```
amount_due = input("Amount due ⇒ ")
paid_money = input("Dollar bills paid ⇒ ")
```

```
if paid_money > amount_due:
    change = paid_money - amount_due
    change % 1 = dollar_bills
    if change ≤ 0.25:
        change - 0.25 = change
        quarters += 1
        print("collect", quarters, "quarters")
    elif change ≤ 0.10:
        change - 0.10 = change
        dimes += 1
        print("collect", dimes, "dimes")
    elif change ≤ 0.05:
        change - 0.05 = change
        nickels += 1
        print("collect", nickels, "nickels")
    elif change ≤ 0.01:
        change - 0.01 = change
        pennies += 1
        print("collect", pennies, "pennies")
```

```
elif paid_money < amount_due:
    change_due = amount_due - paid_money
    print("paid", change_due, "dollars less  
than amount due")
```

```
else:
    print("No change needed.")
```

Write your answers entirely within the boxes below.

---

5. (13 points) Two users play a game by entering strings with English characters only. Write a function `score_calc(word)` that takes a string type as input and returns a score for the word based on the following rules.

Rules for scoring:

- 
1. Even length word with the same first and last letter gets 5 points
  2. Odd length word with the same first , last and middle letter gets 8 points
  3. All a's and e's count towards the score with 1 point for every occurrence.
- 

The player whose string scores more, wins. Then finish up the program by asking each player for their word, and calling `score_calc()` to determine the winner. Your code should handle both upper and lower case. If the user enters nothing or an empty string, then the score should be zero. See a few sample runs below.

```
>>>Player 1 word==>
>>>Player 2 word==>ab
Player 1 score= 0 and Player 2 score= 1
Player 2 won!

>>>Player 1 word==>APala
>>>Player 2 word==>eeae
Player 1 score= 11 and Player 2 score= 9
Player 1 won!

>>>Player 1 word==>eerie
>>>Player 2 your word==>aaria
Player 1 score= 3 and Player 2 score= 3
Its a tie!
```

Write your answer on the next page.

```
player1_word = input("Player 1 word => ")  
player2_word = input("Player 2 word => ")
```

```
def score_calc(word):
```

```
    word.lower()  
    if len(word) % 2 == 0 and word[0] == word[-1]:  
        score += 5
```

```
    elif len(word) % 2 == 1 and word[0] == word[-1]:  
        score += 8
```

```
    elif word.count('a'):  
        score += 1
```

```
    elif word.count('e'):  
        score += 1
```

```
    else:  
        score = 0
```

```
    return score
```

```
score1 = score_calc(player1_word)
```

```
score2 = score_calc(player2_word)
```

```
if score1 < score2:
```

```
    print("Player 2 won!")
```

```
elif score2 < score1:
```

```
    print("Player 1 won!")
```

```
else:  
    print("It's a tie!")
```

6. (14 points) This question is in 2 parts.

**Part a (6/14 points):** In the space below write the contents of the file `line.py`. The file has a function `midpoint`. The function takes a 4-tuple  $(x_1, y_1, x_2, y_2)$  as input with first 2 elements as the  $(x_1, y_1)$  coordinates of a first point and last 2 elements  $(x_2, y_2)$  as the coordinates of a second point. The function returns a 2-Tuple representing the midpoint of the line-segment formed by these points,  $((x_1+x_2)/2, (y_1+y_2)/2)$

Here are a few sample calls for your function:

```
>>> print(midpoint((5,1,6,3)))  
(5.5, 2.0)  
>>> print(midpoint((10,10,30,40)))  
(20.0, 25.0)
```

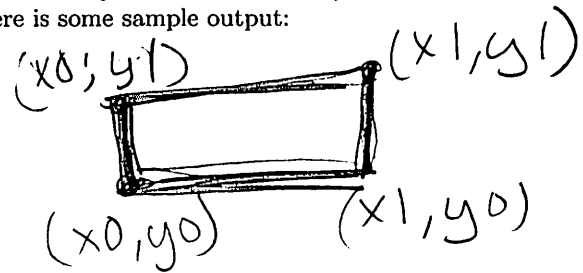
File: `line.py`

```
def midpoint(x1,y1,x2,y2):  
    mid1 = (x1 + y1) / 2  
    mid2 = (x2 + y2) / 2  
    return (mid1, mid2)
```

**Part b: (8/14 points)** Now write code in a separate file called `main.py`. Assume you are given 2 points  $(x_0, y_0)$  and  $(x_1, y_1)$  that are the 2 opposite corners of a rectangle (bottom left and top right). You can assume these are already available to you and you do not need to do input to get the values. Your program should print the midpoints of the 4 sides of the rectangle. For full credit you must use the function written in the `line.py` file. Assume the input is a valid rectangle that has sides parallel to the x and y axes. NOTE: The order in which mid-points are printed does NOT matter. Here is some sample output:

```
>>>x0=0
>>>y0=0
>>>x1=5
>>>y1=10
```

```
First midpoint = (0.0, 5.0)
Second midpoint = (2.5, 0.0)
Third midpoint = (2.5, 10.0)
Fourth midpoint = (5.0, 5.0)
```



```
print("First midpoint= ", midpoint(x0,y0,x0,y1))
print("Second midpoint= ", midpoint(x0,y1,x1,y1))
print("Third midpoint= ", midpoint(x1,y1,x1,y0))
print("Fourth midpoint= ", midpoint(x0,y0,x1,y0))
```

Write your answers entirely within the boxes below.

7. (12 points) Given a 3-Tuple `tup` comprised of three strings, write 3 different single print statement arrangements to print each component of the Tuple on a separate line as follows. For example, if `tup=("Line 1", "Line 2", "Line 3")`, then the output would be

Line 1  
Line 2  
Line 3

All solutions must use the elements of `tup`.

**Part a (3/12 points):** Use 1 print statement with 3 value arguments. Control the new line using the `sep` parameter of `print`.

```
print(tup, sep='\n')
```

**Part b (3/12 points):** Use 1 print statement with a single format string. The `format()` should take the 3 elements of `tup` as arguments.

```
print(tup.format('\n'))
```

**Part c (3/12 points):** Use 1 print statement with a single value argument. Use the concatenation operator and the new line escape sequence to compose a single string.

```
print("tup[0]\n+tup[1]\n+tup[2]\n")
```

**Part d (3/12 points):** Use 3 different print statements.

```
print(tup[0])  
print(tup[1])  
print(tup[2])
```

This page is left blank for scratch work. Do not put your solutions here. Put them in the boxes provided for each question.

**This page is left blank for scratch work. Do not put your solutions here. Put them in the boxes provided for each question.**