LC CS Python

Student Exercise Book

LEAVING CERTIFICATE
COMPUTER SCIENCE

Section 3

Strings

Name: _____

Task 1

Experiment!

A **pangram** is a sentence that uses every letter of the alphabet at least once. Study the program below and see if you can predict what it does? Record your prediction on the right hand side.

Key in the program and make some changes? What happens?

```
1.  # Program to demonstrate basic string operations
2.
3.  # Initialise the string
4.  pangram = "The quick brown fox jumps over the lazy dog!"
5.  #          01234567890123456789012345678901234567890123
6.  # INDEXING
7.  print(pangram[0])
8.  print(pangram[1])
9.  print(pangram[2+4])
10. print(pangram[14])
11. print(pangram[8])
12. print(pangram[43])
13.
14. # The index can be any valid Python expression
15. pos = 17
16. print(pangram[pos])
17. print(pangram[pos+1])
18.
19. # A general pattern used to find the last character
20. print( pangram[len(pangram)-1] )
```

Prediction
Line:
7.
8.
9.
10.
11.
12.
16.
17.
20.

Task 2

Read through the following program. Do you understand what each line of code does?

```
1.  # Program to demonstrate basic string slicing
2.
3.  # Initialise the string
4.  pangram = "The quick brown fox jumps over the lazy dog!"
5.  #          01234567890123456789012345678901234567890123
6.
7.  # Extract from index 1 up to but NOT including index 5
8.  print(pangram[1:5]) # "he q"
9.
10. # Extract from index 17 up to but NOT including index 19
11. print(pangram[17:19]) # ox
12.
13. print(pangram[:19])    # "The quick brown fox"
14. print(pangram[20:26]) # "jumps"
15. print(pangram[26:])    # "over the lazy dog!"
```

What do you think the statement `print(pangram[:])` would display?

Task 3

Q1. Explain what is wrong with each of the following code snippet.
Q2. What do you think the programmer was trying to achieve in the second example?

```
name = "Mary"
print(name[4])
```

```
name = "Mary"
name[3] = "k"
```

Task 4

These programs below illustrate how the plus operator is used to **concatenate** strings.

- Can you predict the output for each program? Make your predictions on your whiteboard.
- Type in each program and get them to run. Did the *actual outputs* match your *predicted outputs*?

PROGRAM 1

```
word1 = "Leaving"
word2 = "Certificate"
word3 = "Computer"
word4 = "Science"
subjectName = word1 + word2 + word3 + word4
print(subjectName)
```

PROGRAM 2

```
pangram = "The quick brown fox jumps over the lazy dog!"
#          0123456789012345678901234567890123456789012345
print(pangram[:3] + pangram[16:])
```

PROGRAM 3

```
noun = input("Enter a singular noun: ")
print("The plural of "+noun+" is "+noun+"s")
```

Log your thoughts.
Each of the above programs contains a deliberate subtle error.
Can you suggest any improvements?

Task 5

The string variable `alNum` is initialised as shown here.

```
alNum = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
#                1         2         3         4         5         6
#       012345678901234567890123456789012345678901234567890123456789012345678901
```

Match each index operation on `alNum` to the correct value in the middle

`alNum[3]`	ABCDEFGHIJKLMNOPQRSTUVWXYZ0
`alNum[52:62]`	d
`alNum[51:62]`	ABCDEFGHIJKLMNOPQRSTUVWXYZ
`alNum[26:53]`	A
`alNum[0:25]`	0123456789
`alNum[1]`	f
`alNum[5]`	b
`alNum[26:52]`	abcdefghijklmnopqrstuvwxyz
`alNum[26]`	Z0123456789

1	2	3	4	5	6	7	8	9

Task 11

The code below initialises four string variables

```
# Initialise four string variables
lowerLetters = "abcdefghijklmnopqrstuvwxyz"
upperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
allLetters = lowerLetters+upperLetters
digits = "0123456789"
```

Match each Python statement to the correct value shown on the right

`print(lowerLetters.upper())`	a
`print(upperLetters.lower())`	Z
`print(upperLetters.find("h"))`	abcdefghijklmnopqrstuvwxyz
`print(lowerLetters.find("h"))`	0123456789
`print(digits.count("123"))`	ABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(lowerLetters.capitalize())`	-1
`print(digits.count("0"))`	1
`print(digits.lower())`	z
`print( min(allLetters) )`	A
`print( max(allLetters) )`	7
`print(allLetters.count("a"))`	0
`print(allLetters.count("aA"))`	abcdefghijklmnopqrstuvwABCABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(allLetters.count("ABC"))`	Abcdefghijklmnopqrstuvwxyz
`print(digits.replace("0", "1"))`	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(allLetters.replace("ABC", "ABC"))`	6
`print(allLetters.replace("xyz", "ABC"))`	1123456789

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Task 11

The code below initialises four string variables

```
# Initialise four string variables
lowerLetters = "abcdefghijklmnopqrstuvwxyz"
upperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
allLetters = lowerLetters+upperLetters
digits = "0123456789"
```

Match each Python statement to the correct value shown on the right

`print(lowerLetters.upper())`	a
`print(upperLetters.lower())`	Z
`print(upperLetters.find("h"))`	abcdefghijklmnopqrstuvwxyz
`print(lowerLetters.find("h"))`	0123456789
`print(digits.count("123"))`	ABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(lowerLetters.capitalize())`	-1
`print(digits.count("0"))`	1
`print(digits.lower())`	z
`print( min(allLetters) )`	A
`print( max(allLetters) )`	7
`print(allLetters.count("a"))`	0
`print(allLetters.count("aA"))`	abcdefghijklmnopqrstuvwABCABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(allLetters.count("ABC"))`	Abcdefghijklmnopqrstuvwxyz
`print(digits.replace("0", "1"))`	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
`print(allLetters.replace("ABC", "ABC"))`	6
`print(allLetters.replace("xyz", "ABC"))`	1123456789

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16