MA1008 Introduction to Computational Thinking Quiz 2.
Answer all the questions in the spaces provided.
AY 2021/2022, Semester 1, Week 9

Your Name: _____ Group: _____

1. Given the following function

```
def doWork(S, n = 1, R = ""):
    if len(S) < 2*n:
        return R
    else:
        return S[:n] + R + S[-n:]
```

State what is the value of `V` after executing the following statements. If a statement is erroneous, state what the error is. (You do not need to provide the correction.) (3 marks each)

i. `V = doWork("A string", R = "=")` `"A=g"`

ii. `V = doWork("A string", 5, "=")` `"="`

iii. `V = doWork(["A", " ", "s", "t", "r", "i", "n", "g"], 3)`

Error. Can't concatenate list and string.

iv. `V = doWork(["A", " ", "s", "t", "r", "i", "n", "g"], 2, ["="])`

`["A", " ", "=", "n", "g"]`

2. A triangular number is a number obtained by summing successive integers starting from 1. For example, 1, 1+2, 1+2+3, 1+2+3+4, etc., produce triangular numbers 1, 3, 6, 10, etc. The function `triangular(n)` takes a positive integer `n` as parameter and returns `True` if `n` is a triangular number and `False` otherwise. Fill in the blanks to complete the function. (12 marks)

```
def triangular(n):
    tri = 0     # holds possible triangular numbers, initialise
    num = 0     # holds successive integers, initialise

    while n > tri:    # continue checking?

        num = num + 1 # get next number

        tri = tri + num # get next triangular number

        if n == tri:    # found triangular number?

            return True

    return False
```

3. i. (a)Tick (✓) the statement(s) below that extend the list `L=[1, 2, 3]` to `[1, 2, 3, 4, 5]`. (4 marks)

1. `L = L + [4, 5]` ✓
2. `L = L.extend([4, 5])`
3. `L.extend([4, 5])` ✓
4. `L.extend(4, 5)`

(b) For each of the statement(s) you did not tick, explain why you did not. (4 marks)

Statement 2 returns L as None, not the required list.

Statement 4 is illegal because the method extend() requires a list as argument.

ii. Given the tuple `T = (1, 2, 3)`, you may extend `T` to `(1, 2, 3, 4, 5)` by doing this: `T = T + (4, 5)`. Explain how this is possible when tuples are immutable. (4 marks)

The tuple on the left of T = T + (4, 5) is a new tuple, not the old tuple extended. So this is not a change of an existing tuple but the creation of a new one, which is allowed.

4. The following program defines the function `task()` that performs some tasks and returns two results. It then calls the function and prints three values. But it contains one or more errors. Identify and correct the error(s) and state what is printed based on your correction. (10 marks)

```
def task(a, b):
    b = a + b
    c = a * b
    a = c * c
    return a, c

a = 2
b = 3
a = task(a, b)
print(a, b, c)
```

The only error is in `print(a, b, c)` where `c` is undefined.
Three possible corrections:
i. Do only `print(a, b)`, then `(100, 10), 3` is printed
ii. Add the statement `global c` at the top of the function.
in which case `(100, 10) 3 10` is printed.
iii. change 2nd last line to `a, c = task(a, b),`
then `100, 3, 10` is printed.

5. i. What does the following statement produce? (5 marks)

```
[a*b for a in "A bus" for b in range(1, 5) if a!=" " and b%2!=0]
```

`['A', 'AAA', 'b', 'bbb', 'u', 'uuu', 's', 'sss']`

ii. Write a list comprehension statement that produces the following list: (5 marks)

```
[16, 9, 4, 1, 1, 4, 9, 16]
```

`[i*i for i in range (-4, 5) if i!=0]`

6. You have created a data base of metals and their properties using lists in Python. The lists are of equal length and the values in the corresponding positions in the lists belong to the same metal. The first five elements of the lists are shown in the statements below. Hence `metal[0]`, `density[0]` and `hardness[0]` all belong to the same metal `"steel"`, with density of 7.75 grams/cm^3 and hardness of 4.5.

```
metal = ["steel", "copper", "zinc", "silver", "gold", …]
density = [7.75, 8.96, 7.13, 10.50, 19.30, …]  # in grams/cm3
hardness = [4.5, 3.0, 2.8, 3.0, 2.5, …]  # on the Mohs scale.
```

Your professor advised that you should instead use a dictionary that has `metal` as the key and a tuple `(density, hardness)` as the value for all the metals present. Write a function called `properties` with the three lists as input parameters and returns the required dictionary, `metal_data`. The three lists already exist and need not be defined in your answer. (12 marks)

```
def properties(metal, density, hardness)
    metal_data = {}
    for i in range(len(metal)):
        metal_data[metal[i]] = (density[i], hardness[i])
    return metal_data
```

7. You have the lists of the first five metals and the densities given in Question 6:

```
metal = ["steel", "copper", "zinc", "silver", "gold"]
density = [7.75, 8.96, 7.13, 10.50, 19.30]  # in grams/cm3
```

plus a new list of the weights of the metals available in stock:

```
weight = [1255.64, 624.85, 70.88, 2404.12, 982.71]  # in grams
```

Write Python statements to print the volumes (in cm^3) of the metals in stock thus:

```
□□□steel□□copper□□□□zinc□□silver□□□□gold
□□162.02□□□69.74□□□□9.94□□228.96□□□50.92
```

Note: volume = weight/density. □ denotes the space character. You may assume that the three lists exist and need not include them in your code. (10 marks)

```
for i in range(len(metal)):
    print(f"{metal[i]:>8s}", end = "")
print()
for i in range(len(metal)):
    print(f"{weight[i]/density[i]:>8.2f}", end = "")
```

8. L is a list of numbers in random order. One method of sorting the numbers in L into ascending order is to take each number, i, in the list in turn, starting from the first, and compare it with every number in the rest of the list. If a number is smaller than i, then the two numbers swap position. Once the first number has been checked against the whole of the rest of the list and the appropriate swaps made, the first item would contain the smallest number in the list. Then it proceeds to check the second, and then the third, and so on, until the end of the list, at which time the whole list would be sorted. The program below performs the sorting and prints the result by defining a function `sort()` with L as the argument.

```
def sort(L):    # function to sort numbers in list L
  for i in range(len(L)):          # visit each item in L
    for j in range(i+1, len(L)): # visit the rest of L
      if L[i] > L[j]: # compare item i with item j
        L[i], L[j] = L[j], L[i] # do the swap the Python way
  return L                         # return the sorted list

L = [40, 1, -21, 5, 32, 86, -73, 9, 24]
print(sort(L))
```

i. Fill in the blanks above to complete the program so that it would print `[-73, -21, 1, 5, 9, 24, 32, 40, 86]`. (6 marks)

ii. Though not an error, the `return` statement in the function is actually not needed. Explain why. (3 marks)

L is a list which is mutable. Hence being the parameter of the function, it carries the new value and is available to the caller. So, no need to return the value separately.

iii. If you remove the `return` statement from the function, how would you modify other parts of the program so that it works correctly? (Only state the modification. No need for the full modified program.) (3 marks)

Since `L` is not returned, the function does not return a value and hence can't perform `print(sort(L))`. Need to first call `sort(L)`, then print `L` thus:

```
sort(L)
print(L)
```

9. i. Create one list each to store the following two matrices: (4 marks)

M1 = $\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$ and M2 = $\begin{bmatrix} 10 & 13 & 16 \\ 11 & 14 & 17 \\ 12 & 15 & 18 \end{bmatrix}$, with each column stored as a list too.

```
M1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
M2 = [[10, 11, 12], [13, 14, 15], [16, 17, 18]]
```

ii. Write a nested `for` loop to perform the matrix sum M3 = M1 + M2 where each element of M3 is the sum of the corresponding elements of M1 and M2. You may initialise the list for the M3 matrix as `[[], [], []]` (6 marks)

```
M3 = [[], [], []]
for i in range(0,3):
    for j in range(0, 3):
        M3[i].append(M1[i][j] + M2[i][j])
```

Or the last statement may be written as

```
        M3[i].extend([M1[i][j] + M2[i][j]])
```

If a student writes the last statement as

```
        M3[i][j] = M1[i][j] + M2[i][j]
```

Award 4 marks as the concept is right, but the execution is wrong, since M3 has not got any j index value yet. A student may initialise M3 as `M3 = [[0,0,0], [0,0,0], [0,0,0]]` then the third statement is OK, but then the first two would not be.

<> <> <> END OF PAPER <> <> <>