SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT133

Timed Online Assignment – January Semester 2020

Structured Programming

Thursday, 14 May 2020 **7:30 pm – 10:00 pm**

Time allowed: 2.5 hours

INSTRUCTIONS TO STUDENTS:

1. This Timed Online Assignment (TOA) comprises **EIGHT (8)** pages (including cover page).

2. You must answer **ALL** questions.

3. If you have any queries about a question, or believe there is an error in the question, while the assignment is in session, briefly explain your understanding of and assumptions about that question before attempting it.

4. You are to include the following particulars in your submission: Course Code, Full Name and Student PI and name your submission file as CourseCode_FullName_StudentPI.

5. Please submit only **ONE (1)** Word file (<500 MB) within the time-limit via Canvas [similar to Tutor Marked Assignment (TMA) submission]. If you do not submit within the time-limit, you would be deemed to have withdrawn (W) from the course. **Appeal is NOT allowed**.

6. To prevent plagiarism and collusion, your submission will be reviewed thoroughly by Turnitin, The Turnitin report will only be made available to the markers. The university takes a tough stance against plagiarism and collusion. Serious cases will normally result in the student being referred to SUSS's Student Disciplinary Group. For other cases, significant marking penalties or expulsion from the course will be imposed.

7. Appendix A is attached.

Answer all questions. (Total 100 marks)

Question 1

Use your understanding of the building blocks of computers and programs to answer this question. You are given the following code segments:

(a)

```
1 a, b = 16, 5
2 c = 0
3 while True:
4     if a >= b:
5         c = c + 1
6         a = a - b
7     else:
8         break
9 print(a, b, c)
```

(i) Describe how a `while` statement and how an `if` statement execute, giving **ONE (1)** difference.

(4 marks)

(ii) Explain in **ONE (1)** sentence what the code segment does.

(4 marks)

(iii) What is the output of the code segment when it is executed?

(3 marks)

(iv) How many times is the statement on line 5 executed?

(3 marks)

(b)

```
1 s = 'abcdefgh'
2 newS = ''
3 for i in range(0, len(s), 3):
4     newS += s[i].upper() + s[i+1:i+3]
5     print(newS)
```

(i) Explain in **ONE (1)** sentence what the code segment does.

(4 marks)

(ii) How many times is the statement on line 3 executed?

(3 marks)

(iii) When the code segment is executed, there is no runtime error. Give the output of the code segment.

(4 marks)

Question 2

Refer to Appendix A should you need the Python documentation.

(a)

```
def larger(a, b):
    return a if a > b else b

def largest(aList):
    theLargest = aList[0]
    for i in range(1, len(aList)):
        if theLargest < aList[i]:
            theLargest = aList[i]
    return theLargest
```

You are given two functions:

- `larger` which takes two numbers as arguments, and returns the larger of the two numbers.
- `largest` which takes a list of numbers as argument, and returns the largest number in the list.

Rewrite the function `largest` so that it calls the function `larger`.

(5 marks)

(b)

```
def largest(aList):
    theLargest = aList[0]
    for i in range(1, len(aList)):
        if theLargest < aList[i]:
            theLargest = aList[i]
    return theLargest

students = {'John': 53, 'Ann': 62, 'Peter': 45, 'Tom': 62}
```

You are given the following Python objects:

- A function `largest` that takes a list of numbers as argument, and returns the largest number in the list.
- A dictionary `students` with name (e.g., John) as key and mark (e.g., 53) as value/

Applying data structures to store and process information, write statements in Python to perform the following tasks:

(i) Print out the highest mark from the dictionary `students`, using the function `largest` in this format: `The highest mark is 62`

(ii) Print the names of students with the highest mark. Each name is on a separate line. The names are preceded by the header: `Top Scorer List`.

```
Top Scorer List
Ann
Tom
```

(iii) Print the number (or the count) of students with the highest mark in this format: `Number of students scoring 62 = 2`

(10 marks)

(c)

```
1  def largest(aList):
2      theLargest = aList[0]
3      for i in range(1, len(aList)):
4          if theLargest < aList[i]:
5              theLargest = aList[i]
6      return theLargest
7  def secondLargest(aList):
8      pass
9
10 aList = [2, 5, 1, 6]
11 print(secondLargest(aList)) # displays 5
12 print(largest(aList)) # displays 6
```

You are given the function `largest` which takes a list of numbers as argument, and returns the largest number in the list. Thus, the statement on line 12 will display 6.

Expressing a sequence of statements based on computational logic, define the body of the function `secondLargest` by replacing the statement `pass` on line 8 with one or more statements.

The function `secondLargest` takes a list of numbers as argument and returns the second largest number in the list such that if the function `secondLargest` is applied on the list, the statement on line 11 will display `5`.

Define the function `secondLargest` to adhere to the following **THREE (3) constraints**:

1. It must call the function `largest`.
2. It should not contain any selection or loop statement.
3. It must not mutate the list.

You will incur a penalty of 5 marks if you violate any of the **constraints**.

(10 marks)

Question 3

This question requires you to solve computational problems using structured programming. Use the functions `randrange` and the built-in function, `round` in your implementation.

`random.randrange(start, stop[, step])`
Return a randomly selected element from range(start, stop, step).

`round(number[, ndigits])`
Return number rounded to ndigits precision after the decimal point. If ndigits is omitted or is None, it returns the nearest integer to its input.

(a) Write a function `getRandomNumber()` that takes **TWO (2)** arguments: a minimum value and a maximum value. The function randomly chooses one integer between the minimum and the maximum values (inclusive of these values), and returns the chosen number.

(3 marks)

(b) Write a function `getRandomAverage()` that takes **THREE (3)** arguments: a minimum value and a maximum value, and a number for how many numbers should be chosen randomly between the minimum and maximum values.

The function calls `getRandomNumber()` repeatedly to get the required number of random numbers, sum them up, and return the average.

(5 marks)

(c) Write a function `checkRandomness()` that takes **FOUR (4)** arguments: a minimum value and a maximum value, the number of random numbers required to compute average, and the number of digits precision after the decimal point to round the average. The function performs these tasks:

- Call `getRandomAverage()` with the minimum and maximum values, and the number of random numbers required, and rounds off the average it obtains from the function call, according to the required precision.
- Compute the average of the minimum and the maximum values, and rounds it off according to the same required precision.
- Display both averages and indicates either the two averages are the same or are different.

Two example outputs where 100 random numbers between 1 and 20 are generated are shown here:

```
Average of 100 random numbers, 10.4 is different from average of 1 and
20, 10.5
Average of 100 random numbers, 10.5 is the same as average of 1 and
20, 10.5
```

(10 marks)

(d) Write an application that performs the following tasks:

- Prompt the user for the minimum and maximum values and the precision.
- Repeatedly prompt for the number of random numbers to test.
  If the number of random numbers to test is not greater than 0, the application ends. Otherwise, the application calls function `checkRandomness()` with the user input.

An example run is shown here, with user input underlined,

```
Enter the minimum value and maximum values, separated by a space: 1 20
Enter number of digits precision after decimal point: 1
Enter number of random numbers to test: 100
Average of 100 random numbers, 11.6 is different from average of 1 and 20,
10.5
Enter number of random numbers to test: 100
Average of 100 random numbers, 10.5 is the same as average of 1 and 20, 10.5
Enter number of random numbers to test: -1
Application ended
```

(7 marks)

Question 4

Employ structured programming principles to develop a program that tracks the claims made for 3 types of car insurance policies, A, B and C.
The program uses a dictionary, `insurance` with policy type as key and a list of claim amounts as value.
An example content of `insurance` is shown here:

```
{'A': [], 'B': [4150, 5502.25], 'C': [4625]}
```

In the example content of the dictionary `insurance`,

- no claim has been made for policy type A,
- two claims with amount $4150 and $5502.25 have been made on policy type B,
- one claim with amount $4625 has been made for policy type C.

The application provides this menu to users:

```
Menu
1. Make A Claim
2. Get Summary
0. Exit
```

(a) Write a function `claim()` for Option 1 with the necessary argument(s), if required. The function prompts for a policy type. If the policy type entered is not A, B or C, the function should print an error message `Invalid policy type`.
Otherwise, the function prompts and reads a claim amount. If the claim amount is 0 or less, print an error message `Invalid claim amount`.
If the claim amount is greater than 0, the function appends the claim amount to the list of claim amounts paid out for that policy type, and prints a message `Successful policy claim`. Append each successful claim as a line containing the policy type followed by the claim amount, into a file, `claims.txt`. An example content for `claims.txt` is shown here:

```
C 1000.0
A 1200.0
C 500.0
```

(8 marks)

(b) Write a function `summary()` for Option 2 with the necessary argument(s), if required. The function prints out a summary of the current content of the dictionary. Refer to the example run for the format of the summary.

(8 marks)

(c) Write a program that repeatedly presents the menu and performs the requested operations until option 0 is chosen. Initialise the dictionary, `insurance` with three policy types A, B and C, each starts with an empty list of claim amount.

(9 marks)

An example run is shown here, with user input underlined:

```
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 1
Enter policy type: D
```

```
Invalid policy type
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 1
Enter policy type: C
Enter claim amount: $1000
Successful policy claim
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 2
Summary of Policies A, B, C
Policy  Total Claimed($)  Number of Claims
  A               0.00                 0
  B               0.00                 0
  C            1000.00                 1
End Summary
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 1
Enter policy type: C
Enter claim amount: $0
Invalid claim amount
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 1
Enter policy type: C
Enter claim amount: $500
Successful policy claim
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 2
Summary of Policies A, B, C
Policy  Total Claimed($)  Number of Claims
  A               0.00                 0
  B               0.00                 0
  C            1500.00                 2
End Summary
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 5
Invalid option
Menu
1.    Make A Claim
2.    Get Summary
0.    Exit
Enter option: 0
Application ended
```

Appendix A

Method	Meaning
<dict>.keys()	Returns a sequence of keys.
<dict>.values()	Returns a sequence of values.
<dict>.items()	Returns a sequence of tuples (key, value) representing the key-value pairs.
<key> in <dict>	Returns true if dictionary contains the specified key, false if it doesn't.
for <var> in <dict>:	Loop over the keys.
<dict>.get(<key>, <default>)	If dictionary has key, returns its value; otherwise returns default.
del <dict>[<key>]	Deletes the specified entry.
<dict>.clear()	Deletes all entries.
<file>.read()	Returns the unread content as a single string
<file>.readline()	Returns the next line of the file.
<file>.readlines()	Returns a sequence (a list) of unread lines in the file.
<file>.write(str)	Writes string to the file, and return the number of characters.
<file>.close()	Closes file and release resources
open(filename, filemode)	Opens file
<list>.append(item)	Adds item at end of list
<list>.insert(pos, item)	Adds item at specified position of list
<list>[pos] = value	Replaces element at pos with value
<list>[start:end] = sequence	Replaces elements at pos start to end -1 with elements in sequence
<list>.remove(item)	Removes item in list
<list>.pop(pos)	Removes item at pos in list
<list>.clear()	Removes all items in list
list(sequence)	Converts sequence to list
<list>.sort()	Sorts (order) the list. A comparison function may be passed as a parameter.
<list>.reverse()	Reverses the list.
<list>.index(x)	Returns index of first occurrence of x.
<list>.count(x)	Returns the number of occurrences of x in list.

----- END OF PAPER -----