NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER I EXAMINATION 2018-2019 SUGGESTED SOLUTION

MH1401 – Algorithms and Computing I

Nov 2018 TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FOUR (4)** questions and comprises **FIVE (5)** printed pages.
2. Answer **ALL** questions. The marks for each question are indicated at the beginning of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This is a **RESTRICTED OPEN BOOK** exam. Each candidate is allowed to bring **ONE (1)** hand-written, double-sided A4 size help sheet.
5. Candidates may use calculators. However, they should lay out systematically the various steps in the workings.

Question 1. **(25 marks)**

(a) What is the value of the following expression?

```
10+10//2**2+5
```

(b) What is printed on the screen when you execute the following commands?

(i)
```
x = 100
y = 99
z = x
x = 'hello'
y = x
print(x,y,z)
```

(ii)
```
print(list(range(3,10)))
```

(iii)
```
print(list(range(3,10,2)))
```

(iv)
```
myList = [10, 20, 'NTU', 'SPMS', 99, 999, 'hi']
ListA = myList[1:5]
ListB = myList[-1:-5:-2]
print(ListA)
print(ListB)
print(ListB*3)
```

(v)
```
L = []
for i in range(10):
    for j in range(10):
        if (i+j)==10 and j>=i:
            L.append([i,j])
print(L)
```

Answer

(a) 10 (`**` has highest order of precedence, followed by `//` and lastly `+`)

(b) (i) ‘hello’, ‘hello’, 100

(ii) $[3, 4, 5, 6, 7, 8, 9]$

(iii) $[3, 5, 7, 9]$

(iv) [20, ‘NTU’, ‘SPMS’, 99]
['hi', 99]
['hi', 99, 'hi', 99, 'hi', 99]

(v) [[1, 9], [2, 8], [3, 7], [4, 6], [5, 5]]

Question 2. **(25 marks)**

(a) Consider the following piece of code:

```
if (age>=12):
    print('You are eligible to see the match.')
    if (age <=20 or age >=60):
        print('Ticket price is $10.')
    else:
        print('Ticket price is $15.')
else:
    print('You are not eligible.')
```

Rewrite the above program so that it does the same thing without using nested `if-else` statements.

(b) Write a program that does the following:

- First, prompt the user for an input of a positive integer which will be assigned to the variable `my_num`. If a user inputs 0 or a negative integer, print an error message, and exit the program.
- With a valid input, generate a list whose length is equal to `my_num`, where each item in the list is a random integer between 1 (included) and 9 (included).
- Then replace every odd integer in the list by 0, and print the resulting list.

Answer

(a)
```
if (age<12):
    print('You are not eligible.')
elif (20<age<60):
    print('You are eligible to see the match.')
    print('Ticket price is $15.')
else:
    print('You are eligible to see the match.')
    print('Ticket price is $10.')
```

*Note: There are other solutions to this question.

(b)

```
import random as rand

my_num = int(input('Input a positive integer: '))
if my_num<=0: print('You have not entered a positive
↪  integer! Exiting ...')
else:
    //Either use for loop or list comprehension

    //1) Using for loop
    L=[0]*my_num
    for i in range(my_num):
        L[i]=rand.randint(1,9)

    //2) Using list comprehension
    L=[rand.randint(1,9) for i in range(my_num)]

    //Replace odd integers
    for i in range(len(L)):
        if L[i]%2==1:
            L[i]=0
print(L)
```

Question 3. **(10 marks)**

```
n = int(input('Enter an integer: '))
num = 0
while (10<=n<=99):
    if n>50:
        break
    num += n
    n = int(input('Enter an integer: '))
else:
    print('ok')
print(num)
```

When the program is run, what will be displayed in the output if the following numbers are entered in the given order at the prompts:

(i) 20, 30, 60

(ii) 20, 30, 40, 7

Answer

(i) 50

(ii) 90

(iii) The number printed on the last line is the sum of the previously entered numbers, excluding the last number (since the previous entered numbers satisfy the *while* condition and **NOT** the nested *if* condition)

Question 4. **(30 marks)**

For this question, you can assume that the NumPy module has already been imported using `import numpy as np`. Besides, you are not allowed to use built-in Python or NumPy functions such as `sum` or `np.average`.

(i) Write a Python function `my_average` that will take as input a matrix `mat` of unknown size, represented as a NumPy two-dimensional array. The function will output the average of the elements of that matrix.

(ii) Write a Python function `is_average` that will take as input a matrix `mat` of unknown size, represented as a NumPy two-dimensional array. The function will out the boolean value *True* if any of the matrix elements is equal to the matrix average, *False* otherwise. You can assume that you have access to `my_average`, the function implemented in the previous question.

(iii) Write a function `sort_average` that will take as input a matrix `mat` of unknown size, represented as a NumPy two-dimensional array. The function will output a list containing all the elements of the matrix, sorted (in increasing order) according to their distance to the matrix average. For an element x and an average value a, the distance is defined as $|x - a|$. You can assume that you have access to `my_average`, the function implemented in the previous question.

For example, with the matrix

$$\begin{bmatrix} 0 & 2 & 0 \\ 2 & 4 & 1 \\ 1 & 5 & 3 \end{bmatrix}$$

the matrix average is 2, and the output of the function `sort_average` should be $[2, 2, 3, 1, 1, 4, 0, 0, 5]$ (some elements could be at a different position as they have the same distance to the average, for example 1's and 3's positions could be permuted).

Answer

(i)
```
def my_average(mat):
    (rows,cols) = mat.shape
    my_sum = 0
    for r in range(rows):
        for c in range(cols):
            my_sum += mat[r,c]
    return my_sum/(rows*cols)
```

(ii)
```
def is_average(mat):
    (rows,cols) = mat.shape
    mat_average = my_average(mat)
    for r in range(rows):
        for c in range(cols):
            if mat[r,c]==mat_average:
                return True
    return False
```

(iii)

```
def sort_average(mat):
    (rows,cols) = mat.shape
    mat_average = my_average(mat)

    my_input_list = []
    for r in range(rows):
        for c in range(cols):
            my_input_list.append(mat[r,c])
    my_output_list = [my_input_list[0]]
    for j in range(1,len(my_input_list)):
        i=0
        while abs(my_output_list[i]-mat_average) <
         ↪  abs(my_input_list[j]-mat_average):
            i += 1
            if i==len(my_output_list):
                break
        my_output_list.insert(i,my_input_list[j])
    return my_output_list
```

END OF PAPER