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.

Solutions provided by: Brandon Goh – bgoh008@e.ntu.edu.sg

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

(iii) In general, describe the number printed on the last line of the output (i.e. how it is related to the numbers entered by the user) when the program terminates.

(iv) Modify the program so that it continues to prompt the user for an input of an integer so long as the previous input is a 2-digit number, i.e. it will terminate only when the integer entered is less than 10 or more than 99. Moreover, the number printed on the last line of the output should be the sum of all the **even** 2-digit integers entered by the user.

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) □

(iv)
```
n = int(input('Enter an integer: '))
num = 0
while (10<=n<=99):
    if n%2==0:
        num += n
    n = int(input('Enter an integer: '))
else:
    print('ok')
print(num)
```

□

Question 4. **(25 marks)**

Assume that the numpy module has already been imported using

```
import numpy as np
```

A **Latin square** is an $n \times n$ array filled with n different integers $x_1, x_2, \cdots, x_n$ such that each of these integers occurs exactly once in each row and exactly once in each column. The (i, j)-entry of a Latin square L, denoted by $L[i, j]$, is the integer that occurs in the cell at row-index i and column-index j, assuming index starts at 0. An example of a 4×4 Latin square L is:

$$\begin{pmatrix} 1 & 6 & 3 & 7 \\ 7 & 1 & 6 & 3 \\ 3 & 7 & 1 & 6 \\ 6 & 3 & 7 & 1 \end{pmatrix}$$

where $L[0, 3] = 7$, $L[1, 2] = 6$ etc.

(i) Write a Python function `is_latin_square` that takes a numpy array as input, and returns the boolean value *True* if it is a Latin square, *False* otherwise. In the case that the input numpy array does not correspond to an $n \times n$ array, the function should return the boolean value *False*.

(ii) An **intercalate** in a Latin square L, denoted by $[i, j, x, y]$, is a pair of rows i and j and a pair of columns x and y such that $L[i, x] = L[j, y]$ and $L[i, y] = L[j, x]$, where $i < j$ and $x < y$. For example, the 4×4 Latin square L above contains an intercalate given by $[0, 2, 1, 3]$ since $L[0, 1] = L[2, 3] = 6$ and $L[0, 3] = L[2, 1] = 7$.
Write a Python function `intercalate` that takes an $n \times n$ Latin square (represented as a numpy array) as input, and returns the following:

(i) the number of intercalates it contains, and

(ii) a list consisting of all the intercalates represented by $[i, j, x, y]$.

Answer

(i)
```
def is_latin_square(mat):
    [r,c] = mat.shape
    if r != c:          #Checks whether it is a square
```

```
        return False
    for i in range(c):
        Ri=mat[i][:]  #Checks uniqueness of row
        if len(np.unique(Ri)) != len(Ri):
            return False
        Ci=mat[:][i]  #Checks uniqueness of column
        if len(np.unique(Ci)) != len(Ci):
            return False
    return True
```

□

(ii)

```
def intercalate(mat):
    store=[]
    [r,c] = mat.shape
    for i in range(r-1):
        for j in range(i+1,r): #Takes care of i<j
            for x in range(r-1):
                for y in range(x+1,r): #Takes care of
                ↪  x<y
                    if mat[i,x]==mat[j,y] and
                    ↪  mat[i,y]==mat[j,x]:
                        store.append([i,j,x,y])
    return len(store), store #Returns number + list
```

□

END OF PAPER