**NANYANG TECHNOLOGICAL UNIVERSITY**

Suggested Solutions (Brandon)

# MH1401/CY1401 - Algorithms and Computing I

November 2016 TIME ALLOWED: 120 MINUTES

---

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FOUR (4)** questions and comprises **SIX (6)** 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 NOT** and **OPEN BOOK** exam.
5. This paper has been converted from the original MATLAB exam to a PYTHON exam. All questions are the property of Nanyang Technological University.

**QUESTION 1.** **(28 marks)**

(a)
```
import math

x = -1
while(x <= 0 or x!=math.floor(x)):
    try:
        x = float(input("Input a positive integer: "))
        if (x <= 0 or x!=math.floor(x)):
            print("The input is not correct!")
    except:
        print("The input is not correct!")

# Alternative solution

x = -1
while(x <= 0 or x!=int(x)):
    try:
        x = float(input("Input a positive integer: "))
        if (x <= 0 or x!=math.floor(x)):
            print("The input is not correct!")
    except:
        print("The input is not correct!")

# the except case is to catch errors if the user inputs a string
```

(b) 0

(c) 18

(d)
```
if x < -3 or x >= 3:
    y = f(x,1)
else:
    if x < 0:
        y = f(x,3)
    else:
        if x == 0:
            y = f(x,4)
        else:
            y = f(x,2)
```

**QUESTION 2.** **(24 marks)**

(i)
```
def income_tax(income):
    if income <= 20000:
        return 0
    elif income <= 40000:
        return 0.05 * income
    elif income <= 100000:
        return 0.1 * income
    elif income <= 200000:
        return 0.15 * income
    else
        return 0.2 * income
```

(ii)
```
def income_tax_sg(income):
    if income <= 20000:
        return 0
    elif income <= 40000:
        return 0.05 * (income-20000)
    elif income <= 100000:
        return 0.05 * 20000 + 0.1 * (income - 60000)
    elif income <= 200000:
        return 0.05 * 20000 + 0.1 * 60000 + 0.15 * \
        (income - 100000)
    else
        return 0.05 * 20000 + 0.1 * 60000 + 0.15 * 100000 + \
        0.2 * (income - 200000)

# The '\' character splits code into multiple lines (making it
# more readable)
```

**QUESTION 3.** **(24 marks)**

(i)
```
def newton_sqrt(X,n):
    if X < 0 or n <= 0:
        return -1;
    if n == 1:
        R1 = 10;
        return R1;
    else:
        out = newton_sqrt(X,n-1)
        Rx = 0.5 * (out**2 + X) / out
        return Rx;

# Note that if they did not ask for recursion,
# we can use the following for loop instead:

# Non-recursion method:
def newton_sqrt(X,n):
    R1 = 10
    for i in range(1,n+1):
        if i-1 == 0:
            Rx = R1
        else:
            Rx = 0.5 * (R1**2 + X) / R1
            R1 = Rx
    return Rx
```

(ii)
```
def newton_sqrt_approx(X,a):
    import math
    i = 0;
    dist=a+1;
    while (dist > a):
        i+=1;
        temp = newton_sqrt(X,i)
        if temp == -1:
            return (-1,0)
        else:
            dist = abs(math.sqrt(X)-(newton_sqrt(X,i)))
    return (i,dist)

"""
The above implementation works by adding these lines:

Part (i):
```

```
x=int(input("Input a non-negative number X: "))
n=int(input("Input a positive number n: "))
print("Result = %.4f" % newton_sqrt(x,n))

Part (ii):
x=int(input("Input a non-negative number X: "))
a=float(input("Input a positive number a: "))
(n,dist) = newton_sqrt_approx(x,a)
if n == -1:
    print("Invalid input(s)!")
else:
    print("%d iterations required, distance = %.6f" % \
(n,dist))
"""
```

**QUESTION 4.** **(24 marks)**

(i)
```
def check_move(x,y):
    if x == []:
        return -1
    elif y== []:
        return 0
    elif x[-1] > y[-1]:
        return -1
    else:
        return 0
```

(ii)
```
def check_victory(x):
    if len(x) != 5:
        return 0
    else:
        for i in range(len(x)-1):
            if x[i] <= x[i+1]:
                return 0
    return 1
```

(iii)
```python
vectorofrods = [[5,4,3,2,1],[],[]]
win = 0;
while(win != 1):
    movevalid = -1
    while (movevalid != 0):
        movefrom = int(input( \
        "Which rod do you want to move the disc from? "))
        moveto = int(input( \
        "Which rod do you want to move the disc to? "))
        movevalid = check_move(vectorofrods[movefrom],\
        vectorofrods[moveto])
        if (movevalid != 0):
            print("Invalid move, try again!\n")
            print("Current rod list:", vectorofrods)
        else:
            vectorofrods[moveto].append( \
            vectorofrods[movefrom].pop())
            print("Current rod list:", vectorofrods)
    win = check_victory(vectorofrods[moveto])
print("Congratulations to you for finishing MH1401!")

"""
Part (i), (ii), (iii) is a working implementation of
Tower of Hanoi. You can play it by copying the code
into Spyder and running it!

Good luck! :)
"""
```