NANYANG TECHNOLOGICAL UNIVERSITY

Suggested Solutions (Camille)

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)
```
check = 0
while check == 0:
    try:
        x = int(input('Please enter a positive integer x: '))
    except ValueError:
        print('Error: Not an integer!')
    else:
        if x > 0:
            check = 1
        else:
            print('Error: Non-positive integer detected!')

"""
The above code implementation loops the input request
until a positive integer is entered.

If only a single input is desired, simply add in check = 1
just before the try statement.
"""
```

(b) False

(c) 18

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

QUESTION 2. **(24 marks)**

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

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

```
"""
Note: \ is used as the linebreak operator.
It's basically just to split up long lines,
but in the code it's treated as the same line.
"""
```

QUESTION 3. **(24 marks)**

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

(ii)
```
import math

def newton_sqrt_approx(X,a):
    if newton_sqrt(X,1) == -1
        print('Error in X value')
        return (1,-1)
    result = a + math.sqrt(X) + 1
    counter = 0
    while abs(result - math.sqrt(X)) > a:
        counter = counter + 1
        result = newton_sqrt(X,counter)

    return (counter, abs(result - math.sqrt(X)))
```

QUESTION 4. **(24 marks)**

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

(ii)
```
def check_victory(x):
    if x == [5,4,3,2,1]:
        return 1
    else:
        return 0
```

(iii)
```
rods = [[5,4,3,2,1],[],[]]
check = check_victory(rods[2])
while check == 0:
    print('Please make the next move.')
    xind = int(input('Input the index of rod x: '))
    yind = int(input('Input the index of rod y: '))
    x = rods[xind]
    y = rods[yind]

    if check_move(x,y) == 0:
        y.append(x[len(x)-1])
        x.pop()
        rods[xind] = x
        rods[yind] = y
        print('Move made. Below is the current game state.')
    else:
        print('Invalid move. Below is the current game state.')

    check = check_victory(rods[2])
    print(rods)

print('Congratulations, you win!')
```