NANYANG TECHNOLOGICAL UNIVERSITY

Suggested Solutions

MH1401/CY1401 - Algorithms and Computing I

<u>NOTE:</u>

1. The following paper has been converted from MATLAB to Python.

Contents

QUESTION 1. **(28 marks)**

(a)
```
x=-1
while(x<=0):
    x=int(input("Input a positive integer"))
```

(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;
end
```

(ii) In Singapore (and in many other countries), the rule is slightly more complex as the income is taxed in layers, with a higher tax rate applied to each successive layer. Using the same Table as before, a citizen will pay a 0% tax rate for its first 20,000 SGD, then a 5% tax rate for the next 20,000 SGD, then a 10% tax rate for its next 60,000 SGD, etc.

For example, if a citizen has an income of 145,000 SGC, the first 20,000 SGD are taxed at a 0% tax rate, then the nexr 20,000 SGD are taxed at a 5% tax rate, then the next 60,000 SGD are taxed at a 10% rate, and finally the remaining $45,000$ SGD are taxed at a 15% rate. In total, he would have to pay $(20,000 \times 0 + 20,000 \times 0.05 + 60,000 \times 0.1 + 45,000 \times 0.15) = 13750$ SGD.

Write again a function `income_tax_sg` that will take as input the income of the citizen, and that will return the income tax amount he has to pay for this new tax system.

QUESTION 3. (10 marks)

Newton's method is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function. It can be used to easily find a good approximation of the square root of a number $X \geq 0$. Let $R_1 > 0$ be a rather close approximation of $\sqrt{X}$, then $R_2 = \frac{1}{2}\left(\frac{R_1+X}{R_1}\right)$ offers an even better approximation of $\sqrt{X}$.

(i) Write a **recursive** function `newton_sqrt(X,n)` that will return the n-th approximation of $\sqrt{X}$ using Newton's method (starting with $R_1 = 10$ as first approximation). As error check, the function returns -1 when X is negative or when n is not a positive integer.

(ii) Assume that you have access to the function `newton_sqrt(X,n)` described above. Write a function `newton_sqrt_approx(X,a)` that will output

- how many approximation steps are needed using Newton's method (starting with $R_1 = 10$ as first approximation), so that the distance between the approximation and the real $\sqrt{X}$ value is smaller of equal to a, and
- the corresponding distance value when the sufficiently close approximation is found.

Warning: note that the function outputs two values (by output, we mean that the function itself outputs the value, not just a printing on the screen). Hint: you can use the built-in functions `sqrt` and `absolute` in the `numpy` package in PYTHON.

QUESTION 4. (10 marks)

The Tower of Hanoi is a well-known mathematical game. It consists of **three rods**, and a number of disks of different sizes which can slide onto any of the three rods. The puzzle starts with all the disks stacked in ascending order of size on the first rod, the smallest at the top, thus making a conical shape (see picture below). The objective of the puzzle is to move the entire stack to the third rod (only one disk can be moved at a time), obeying the following simple rules:

- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack, i.e. a disk can only be moved if it is the uppermost disk on a stack.

- No disk may be placed on top of a smaller disk.

(i) Write a **recursive** function `newton_sqrt(X,n)` that will return the n-th approximation of $\sqrt{X}$ using Newton's method (starting with $R_1 = 10$ as first approximation). As error check, the function returns -1 when X is negative or when n is not a positive integer.

(ii) Assume that you have access to the function `newton_sqrt(X,n)` described above. Write a function `newton_sqrt_approx(X,a)` that will output

- how many approximation steps are needed using Newton's method (starting with $R_1 = 10$ as first approximation), so that the distance between the approximation and the real $\sqrt{X}$ value is smaller of equal to a, and
- the corresponding distance value when the sufficiently close approximation is found.

Warning: note that the function outputs two values (by output, we mean that the function itself outputs the value, not just a printing on the screen). Hint: you can use the built-in functions `sqrt` and `absolute` in the `numpy` package in PYTHON.

Suggested Solutions (Brandon)

Suggested Solutions (Camille)