NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2016-2017

MH1401/CY1401 - Algorithms and COmputing I

August 2017 TIME ALLOWED: 30 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) Assume that an integer input `x` is given by a user. How do you check that `x` is a positive integer?

(b) Given a vector `vec` initialised as $[1, 0, 3, 0]$, what would be the result of the following command?

```
any(vec) && all(vec)
```

(c) Given the matrix

$$\texttt{mat} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

what is the result of the following command?

```
np.sum(mat[0,2]*mat[0,:]))
```

(d) Rewrite the following `if-elif` statement as a nested `if-else` statement that accomplishes exactly the same thing. Assume that `x` is an integer variable that has been initialised and the function `f(x,d)` is defined.

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

QUESTION 2. **(24 marks)**

In Singapore, personal income tax rates for resident taxpayers are progressive. This means higher income earners pay a proportionately higher tax, with the current highest personal income tax rate at 20%.

(i) Imagine a very simple tax system where a citizen pays an income tax rate with the rule from the table below.

Income (in SGD)	**Tax rate**
0 to 20,000 included	0%
20,001 to 40,000 included	5%
40,001 to 100,000 included	10%
100,001 to 200,000 included	15%
more than 200,001	20%

For example, if a citizen has an income of 45,000 SGD, he will be in the 10% rate section, so he will pay $45,000 \times 0.1 = 4,500$ SGD. Write a function `income_tax` that will take as input the income of the citizen, and that will return the income tax amount he has to pay. Assume that the income value input is always an integer.

(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 SGD, the first 20,000 SGD are taxed at a 0% tax rate, then the next 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 $(20,000 \times 0 + 20,000 \times 0.05 + 60,000 \times 0.1 + 45,000 \times 0.15) = 13,750$ 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. Assume that the income value input is always an integer.

QUESTION 3. **(24 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

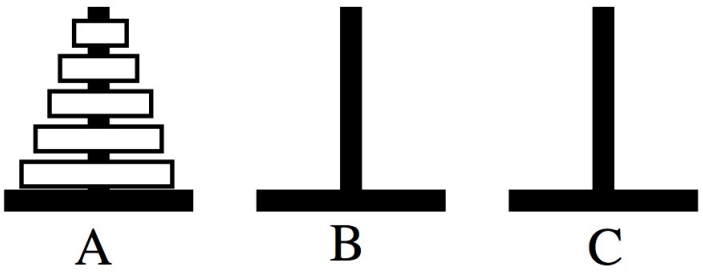
- how many approximation steps are needed using Newton's method (starting with $R_1 = 10$ as first approximation), so that the distance between the approximation and the real $\sqrt{X}$ value is smaller of equal to a, and
- the corresponding distance value when the sufficiently close approximation is 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. **(24 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.



One can model this game in PYTHON by representing each rod by a vector (say `a` for first rod, `b` for second rod, `c` for third rod), and each disk by a value (a bigger disk is represented by a bigger value). The first element of a rod vector represents the closest disk to the base of the rod, then the second element of a rod vector represents the second closest disk to the base of the rod, etc. For example, with five disks, at the start of the game, `a` would be the vector $[5, 4, 3, 2, 1]$, while `b` and `c` would be empty vectors. If I move the top disk from the first rod to the third rod, then `a` would be the vector $[5, 4, 3, 2]$, while `b` would be an empty vector and `c` would be the vector $[1]$.

Note: Question No. 4 continues on page 6

(i) Implement a function `check_move` that takes as input two rod vectors `x` and `y`. The function will check if moving the top disk of rod `x` to rod `y` is a valid move (according the game rules). Note that if rod `x` is empty, it is also considered an invalid move. If the move is invalid, the function will return the value -1. Otherwise, if the move is possible, the function will return 0.

(ii) Implement a function `check_victory` that takes as input a rod vector `x`, and outputs the value 1 if this rod contains all the disks (you can assume that the game contains 5 disks in total). Otherwise, it outputs 0.

(iii) Assume that you have access to the functions `check_move` and `check_victory` described above. Implement a script that will ask a human player to input two rods' indexes `x` and `y` (you can assume that the player never inputs an error, the indexes always belong to $[1, 2, 3]$), check if the move from red `x` to rod `y` is valid, and execute this move if it is indeed a valid one. The script will then repeat the process until a victory is attained by the human player. (Hint: use a list or an array to store the three rods' vectors, so you can access them directly with indexes.)