CSC1005: Introduction to Computer Engineering Programming and Applications Assignment 4

Assignment description:
This assignment will be worth 16% of the final grade.

You should write your code for each question in a .py file (please name it using the question name, e.g. q1.py). Please pack all your .py files into a single .zip file, name it using your student ID (e.g. if your student ID is 123456, then the file should be named as 123456.zip), and then submit the .zip file via Blackboard.

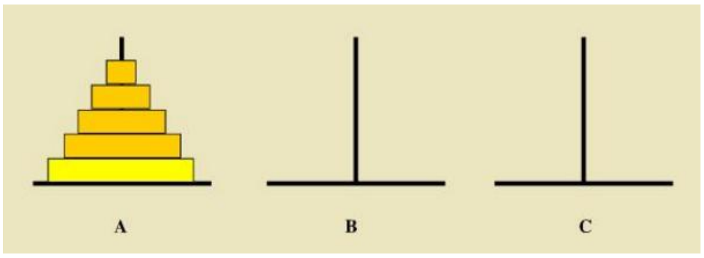
Please create a Word document that provides a detailed explanation of the code for each question. This document should clearly outline the logic behind your code, specify the expected inputs, and describe the anticipated outputs. Make sure the explanations are easy to understand and provide enough context for anyone reviewing your work. Include this Word document in the .zip file along with your code.

Please be aware that the teaching assistant may ask you to explain your code to verify that it was written by you. Additionally, your submission may be checked for similarities with other students' work using Blackboard to ensure academic integrity.

This assignment is due on 5:00PM, 14 Dec (Sunday). For each day of late submission, you will lose 10% of your mark in this assignment. If you submit more than three days later than the deadline, you will receive zero in this assignment.

If you have any questions about Assignment 4 (AS4), please contact the TA for AS4, GUI Xuanang, at 224010127@link.cuhk.edu.cn.

Question 1 *(30% of this assignment):*
The Tower of Hanoi is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The following figure shows the initial state of the Tower of Hanoi with 5 disks.



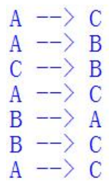
The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. 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.
3. No disk may be placed on top of a smaller disk.

Assume that initially all the disks are placed on rod A. Write a non-recursive Python function to print out the steps to move all the disks from rod A to rod C via rod B (Hint: a recursive algorithm can be converted into a non-recursive algorithm using stack). The header of the function is:

```
def HanoiTower(n)
```

Here n represents the number of disks. For example, when n = 3 your function should output:



```
A --> C
A --> B
C --> B
A --> C
B --> A
B --> C
A --> C
```

Question 2 (*30% of this assignment*):
The Binary Tree is a fundamental data structure in computer science. A binary tree consists of nodes where:

1. Each node contains a value.
2. Each node can have at most two children: a left child and a right child.
3. The topmost node is called the root, and nodes with no children are called leaves.

Binary trees are often used in applications like searching, sorting, and parsing expressions. One of the most common ways to traverse a binary tree is Depth-First Search (DFS), which can be done in three steps recursively:

1. Preorder (Root → Left → Right): Visit the root, then recursively visit the left and right subtrees.
2. Inorder (Left → Root → Right): Recursively visit the left subtree, then visit the root, and finally the right subtree.
3. Postorder (Left → Right → Root): Recursively visit the left and right subtrees, then visit the root.

For this question: You are required to write a Python program to perform Depth-First Search (DFS) traversal on a binary tree.

The binary tree will be represented as a list, where the root node is at index 0, the left child of a node at index i is located at index 2*i + 1, and the right child is located at index 2*i + 2. Empty nodes are represented by None in the list.

Example:
If the binary tree is represented as:

```
tree = [1, 2, 3, 4, 5, None, 6]
```

The tree structure is:

```
      1
     / \
    2   3
   / \   \
  4   5   6
```

Requirements:

1. Write a Python function dfs_traversal(tree, order) that performs DFS traversal on the input binary tree.
   - ✓ The parameter tree is a list representation of the binary tree.
   - ✓ The parameter order specifies the traversal order ("preorder", "inorder", or "postorder").
   - ✓ Return the list of values visited during the traversal.
2. Test the function with the example tree above for all three traversal orders.

Example Output:

```
tree = [1, 2, 3, 4, 5, None, 6]

print(dfs_traversal(tree, "preorder"))  # Output: [1, 2, 4, 5, 3, 6]
print(dfs_traversal(tree, "inorder"))   # Output: [4, 2, 5, 1, 3, 6]
print(dfs_traversal(tree, "postorder")) # Output: [4, 5, 2, 6, 3, 1]
```

Notice: We will follow the above test code to check your function.
Input:
[1, 2, 3, 4, 5, None, 6]

The OJ pretest will output:
[1, 2, 4, 5, 3, 6]←[4,2,5, 1, 3, 6]←[4,5,2,6,3, 1]
Do not write `print()`in your code.

Question 3 *(40% of this assignment)*

The Power System is a network of electrical components used to generate, transmit, and distribute electric power. A key challenge in power system operation is determining how to allocate the required load demand among multiple power generators while minimizing the total generation cost. This problem is known as the Economic Dispatch Problem.

In the Economic Dispatch Problem, the objective is to determine the optimal power output of each generator to meet a given total load demand while minimizing the total cost of generation. Each generator has a cost function that models its fuel cost as a function of the power it generates. These cost functions are often quadratic in nature and take the form:

$$C_i(P_i) = a_i P_i^2 + b_i P_i + c_i$$

where:

- $C_i(P_i)$ is the cost of generating power P_i (in \$/MW).
- a_i, b_i, and c_i are coefficients specific to generator i.
- P_i is the power output of generator i (in MW).

Problem Description:

You are tasked with solving the economic dispatch problem for a power system consisting of **3 generators**. The cost functions for the generators are as follows:

1. Generator 1: $C_1(P_1) = 0.02P_1^2 + 10P_1 + 100$
2. Generator 2: $C_2(P_2) = 0.03P_2^2 + 8P_2 + 120$
3. Generator 3: $C_3(P_3) = 0.025P_3^2 + 9P_3 + 150$

The total load demand P_{load} is **300 MW**, and the generators must meet this demand collectively:

$$P_1 + P_2 + P_3 = P_{\text{load}}$$

Additionally, each generator has minimum and maximum power generation limits:

- Generator 1: $50 \leq P_1 \leq 150$
- Generator 2: $50 \leq P_2 \leq 100$
- Generator 3: $50 \leq P_3 \leq 120$

Your Task:

1. Write a Python program to solve the economic dispatch problem and find the optimal power outputs P_1, P_2 *and* P_3 that minimize the total generation cost while meeting the load demand and satisfying the constraints.
2. The program should compute:
   - The optimal power output of each generator.
   - The total generation cost.

Requirements:

- Use numpy to handle matrices and arrays.
- Use scipy.optimize.minimize for constrained optimization.

Tips:

- Formulate the optimization problem as minimizing the total cost:

$$C_{\text{total}} = C_1(P_1) + C_2(P_2) + C_3(P_3)$$

- Include the constraint $P_1 + P_2 + P_3 = P_{\text{load}}$.
- Include the bounds on P_1, P_2, and P_3.

Example Outputs:

Your program should output:

```
Optimal Power Outputs:
P1: <value in MW>
P2: <value in MW>
P3: <value in MW>

Total Generation Cost: <$value>
```

Notice for OJ: Round your outputs as an integer!