University of Science and Technology of Hanoi ******* **Final Examination** **Subject: Algorithms and Data Structures** Sheet: 01 No of pages: 02		***Academic year**: 2023–2024* ***Date**: 09/11/2023* ***Time**: 70 minutes* ***Important instructions*** *(according to lecturer's decision)* 1. Only the course slides and your own exercise code are allowed in the examination venue. 2. Copy or using the Internet will lead to heavy penalty.	
Pathway coordinator		**Lecturer (or Head of Subject)**	**Dr. Đoàn Nhật Quang**
Student name		**Student's ID**	

Follow this instruction:

- Create a folder "YOURNAME_STUDENTID" in the Desktop (STUDENTID = BI12-123, BA11-999)

*- Create the source files **question1.c** (or cpp) and **question2.c** for the corresponding problems.*

*- **Remove the executable files** (.exe), **zip** all your source codes, and submit to the Google classroom, in "**Final Exam Session 1**":* https://classroom.google.com/u/1/c/NjIxMzQ0MzcwNzU5

- Verify your name in the files and mails, un-named or incorrect-name files lead to 0.

Problem 1 (11pts)**:**

Initialize a random array of your choice. The current goal is to verify whether an array number is **centered cube**.

Note: a centered cube number is calculated by the following formula:

$$f(n) = n^3 + (n+1)^3$$

For example, the centered cube numbers are:

f(1) = $1^3 + (1+1)^3 = 9$

f(2) = $2^3 + (2+1)^3 = 8 + 27 = 35$

f(3) = $3^3 + (3+1)^3 = 27 + 64 = 91$

…..

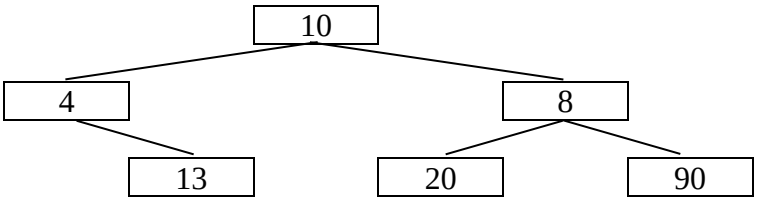
- Write pseudo-code using **Recursion** (if necessary, Iteration) to check all values in the given array are centered cube or not. (2pts)
- Write a program in C/C++ to complete your proposed algorithm in the pseudo-code. (7pts). Note: The program must include a recursive function and the main function.
- Calculate the complexity of your algorithm. Justify the answer. (2pts)

Problem 2: (2pts)

This problem requires to traverse and display a tree using the following process:

- Visit a node and find its child nodes.
- Put the child nodes into the queue.
- Pop a node in the queue and repeat the process until all nodes are visited.

Justify and print the traversal result for the below tree.



Problem 3: (7pts)

Given an array of n elements, the **Bubble sort** algorithm swaps repeatedly the adjacent elements if they are in the wrong order (ascending order). We want to implement an improve version of this sorting algorithm.

The idea is to sort both maximum (at the end) and minimum (at the beginning) in one pass and the array is sorted from both ends.

- Traverse from left to right and compare adjacent elements and the maximum is placed at right side.
- Traverse from right to left and compare adjacent elements and the minimum is placed at left side.
- Repeat the process until the array is sorted.

Implement this algorithm in C/C++. Calculate the complexity of the above algorithm. Justify your answer.

--END--