VNUHCM-UNIVESRITY OF SCICENCE

FINAL EXAMNINATION

Semester III – Academic year 2021-2022

ARCHIVE CODE
(written by ET&QA Office)
CLC_CK21223_CSC10004

Course name: Data Structures and Algorithms(21CLC02) **Course code:** CSC10004

Time: 120 minutes Date: 08/08/2022

Note: *Students are* [☐ *allowed* / ☒ *not allowed*] *to use electronic devices during the examination*

Full name of Student: ..... **Student ID:** ..... **No:** .....

Question 1 (20 points).

Consider the following algorithm:

```
ALGO_A(X) {
    d = ∞;
    for (i = 1; i ≤ X.length - 1; i++)
        for (j = i + 1; j ≤ X.length; j++)
            if (|X[i] - X[j]| < d)
                d = |X[i] - X[j]|;
    return d;
}
```

a. Interpreting `X` as an array of coordinates of points on the x-axis (horizontal axis), explain concisely what `ALGO_A(X)` does, and give a tight asymptotic bound for the complexity of `ALGO_A(X)`.

b. Write an algorithm `BETTER_A(X)` that is functionally equivalent to `ALGO_A(X)`, but with a better asymptotic complexity. Show the complexity of `BETTER_A(X)`.

Question 2 (20 points).

Given a circular sorted array of n integers, write a C/C++ function (and auxiliary functions if needed) running in $O(\log n)$ to search an element in it. Assume that there are no duplicates in the array and the rotation is in counterclockwise direction.

The prototype of the function is as follows:

```
int searchArray(int a[], int n, int k);
```

where `k` is the value to be searched for and the return value of the function is the index of the element whose value is `k` or -1 if the element is not present in the input array.

Example:

Input array: 7 9 11 1 3 5
Search value: $k = 3$
Output: 4

Input array: 9 11 1 3 5 7
Search value: $k = 11$
Output: 1

Input array: 3 5 7 9 11 1
Search value: $k = 10$
Output: -1

Question 3 (30 points).

Given a binary tree, you are asked to write a C/C++ function (and auxiliary functions if needed) to display the keys of all the nodes in the longest path going from the root to a leaf of the tree. If there are many such paths, display any of them. The prototype of the function is as follows:

```
void printLongestPath(Ref r);
```

Question 4 (30 points).

Initially, assume that the given AVL tree is empty. You are asked to expand the tree by inserting keys (one at a time) to it. Draw the tree after each node insertion. If the tree is imbalanced at a node, rebalance the (sub)tree rooted at that node and draw the resulting tree. The expanding process takes place continuously until all of four imbalanced cases: LL, RR, LR, and RL have been occurred.

Note:

- The keys to be inserted to the tree are positive integers selected arbitrarily.
- Rebalancing a (sub)tree with only three nodes is not counted.
- Do not use the example shown in class.