ST. CHARLES
COMMUNITY COLLEGE

CPT-281 - Introduction to Data Structures with C++

Sample Exam 2 (40 Points)

Part I - Multiple-Choice Questions (16 Points)

- There are 8 multiple-choice questions (Questions 1 to 8) in this part. Each question's value is 2 points.
- In each question, there exists only one correct/best answer.
- For each question, your score will be either 0 (you did **not** select the correct answer) or 2 (you selected the correct answer). No partial credits will be given in this part.

1. On average, what is the Big-O of accessing an element based on its key in a hash-table?

   A. $O(\log n)$

   B. $O(n)$

   C. $O(1)$

   D. $O(n \log n)$

2. Which of the following sorting algorithms is based on recurrence relations?

   A. Heap sort

   B. Insertion sort

   C. Merge sort

   D. Radix sort

3. In AVL trees, how many rotations do we need to balance a right-right tree?

   A. 0

   B. 1

   C. 2

   D. 3

4. The most efficient way to resolve collision in a hash table is to ___________.

   A. linear open addressing

   B. quadratic open addressing

   C. chaining

   D. (None of the above)

5. Which of the following is an application of binary search trees?

   A. Database query

   B. Undo/redo system

   C. Depth-first graph traversal

   D. Print job

6. The Big-O of searching for an item in a binary search tree is ___________.

   A. $O(\log n)$

   B. $O(n)$

   C. $O(h)$

   D. $O(n \log n)$

7. What is the advantage of heap sort over merge sort?

   A. Heap sort is $O(n)$ while merge sort is $O(n \log n)$.

   B. Less usage of memory

   C. A stable sort

   D. (None of the above)

8. On removing an item with two children from a binary search tree, the replacement for the item can be ___________.

   A. The postorder predecessor

   B. The inorder successor

   C. The preorder successor

   D. The preorder predecessor

Part II - Data Structure Question (6 Points)

- There is 1 question (Question 9) in this part. The question's value is 6 points.
- The question in this part is **not** a programming question. You **must** follow the question's requirements to answer the question.
- Your answer must be readable and understandable without extra unnecessary efforts.
- In this part, if your answer is **not** 100% correct, partial credits may be awarded based on the quality of your answer.

9. Build an AVL tree from the following integers: 1, 18, 14, 16, 19, 30 (in that order). Your answer **must** be stepwise, e.g., you need to draw the AVL tree after inserting 1, the AVL tree after inserting 18, and so on.

Part III - Programming Question (8 Points)

- There is 1 question (Question 10) in this part. The question's value is 8 points.
- In this part, you need to write some C++ code to answer the question (solve the problem).
- Your code **must** be readable and understandable without extra unnecessary efforts.
- In this part, partial credits may be awarded based on the quality of your code.

10. C++ struct **Tree_Node** is defined as below:

```cpp
struct Tree_Node {
    int val;
    Tree_Node *left, *right;
    Tree_Node(int val = 0, Tree_Node* left = NULL, Tree_Node* right = NULL):
        val(val), left(left), right(right) {}
};
```

Given a binary tree of integers, please write a recursive function that writes the balance number of each node to the console.

- The balance number of a node is defined as the height of its right subtree minus the height of its left subtree.
- Initially, you only have a pointer to the root node of the binary tree.
- Your function can write the balance numbers of the nodes in any order, but each node's balance number should be written only once.
- Your algorithm should have time complexity of $O(n)$, where n is the number of nodes in the binary tree. Failing to design an algorithm with required time complexity will result in losing at least 50% of points. You are **required** to design an efficient algorithm, **not** just designing a correct algorithm.
- Please only complete the required function. Do **not** write a **main()** program.

```cpp
class Solution {
public:
    /** Writes the balance number of each node to the console.
        @param root: root node of the binary tree
        @return: height of the binary tree
    */
    static unsigned int balance_numbers(Tree_Node* root) {
        // Please copy this code to start answering the question.
        // Please add your code to solve the problem.
    }
};
```

Part IV - Algorithm Questions (10 Points)

- There is 1 question (Question 11) in this part. However, the question has multiple parts.
- In this part, you **must** present your algorithms using structured language (e.g., pseudocode). Writing paragraphs or drawing flowcharts to present algorithms will result in 0 points for the question.
- Your algorithms **must** be readable and understandable without extra unnecessary efforts.
- In this part, partial credits may be awarded based on the quality of your answer.

11. A bookstore keeps track of the books it sells. Each book receives reviews from customers who bought the book. The bookstore wants to build a system that keeps track of the books and their reviews. The system should help the staff find a book efficiently using its ID (e.g., ISBN number). Furthermore, the system should allow the staff to query the reviews a book received. For instance, it should be efficient to find all the reviews a book has received between March 1, 2024, and March 31, 2024.

    The following assumptions are made for the system:

    - A book has an ID (e.g., ISBN) and a title.
    - A review has a rating (scaling from 1 to 10) and a date.
    - You may assume that the date is unique for each review.

    1) (2 pts) Which abstract data structure would you use to help the system keep track of books? Defend your answer.

    2) (2 pts) Which abstract data structure would you use to help a book keep track of the history of its reviews? Defend your answer.

    3) (6 pts) Write a pseudo-code algorithm to find the number of reviews after a given date.