(2024-25 CS2DSA) Data Structures and Algorithms in Java

CS2DSA January Exam 2023-24

This is a **closed-book**, timed assessment (**Time allowed: 2 hours**, unless you are entitled to extra time for your assessments). Once started, you must complete this assessment in one sitting.
Answer ALL questions. (100 marks)
See the **Case Study** descriptions for details about each case study.
Once you have started the quiz using the **Respondus lockdown browser**, you must reamin in the Blackboard quiz page throughout the exam duration. Make sure that all your answers are saved. **Do not attempt to leave the Blackboard quiz page before you have submitted your answers.** Leaving the quiz page will terminate the Blackboard quiz session and you will *not* be able to resume your quiz even if the time has not yet run out.

During the assessment, if you try to navigate to another page within Blackboard, e.g. "Learning Resources", the browser will prompt you to confirm:
"Is it OK to leave/reload this page?"
If you choose **"OK**", you will *leave* the quiz page as you have confirmed that you want to leave. Once you have left the quiz page, you are considered to have completed your CS2DSA Exam and will not be able to return to the quiz page.
If you choose " **Cancel**", you will be able to continue with your quiz.

Read the questions carefully before you answer them. **Save your answer** after answering **each** question.

Be very careful with your **spelling** for the **textual answers**. Mis-spelt answers may be regarded as incorrect and no mark will be awarded.

To see a record of which questions you have answered and for a quick way to move between questions, click on the **small arrow** to the **left** of **Question Completion Status**.

When you have completed the quiz, click the Submit button. You must then click OK in another window to confirm your submission.

Question 1

6 points

Which of the following problems are best-suited to be solved using the Abstract Data Type (ADT) **tree**?
Choose only the answers you are confident in. Wrong answer choices will lead to mark deduction that will not let your question score go below 0.

Choose at least one correct answer

A Managing the precedence of operations in a mathematical expression.

(B) Maintaining a record of customer transactions in a banking system in the order in which they occur.

(C) Storing the management structure of an organisation in which each employee is managed by one manager.

(D) Facilitating "undo" and "redo" operations in a word processor.

(E) Storing a sorted list of numbers to facilitate efficient search, insertion, and deletion operations.

Question 2

4 points

Consider the following sequence of numbers stored in an `int` array (i.e. `int[]` as defined in Java):
2, 5, 3, 8, 6, 9, 1, 4, 7
These numbers are stored in the array in the order as listed above, i.e., the number '2' at index 0, '5' at index 1, etc.
Imagine that you need to find the number '7' and, if it is not in the array, insert it in a position to maintain the array's order. Which of the following pairs of algorithms would be the most efficient to determine:

- whether the number '7' is in the array, and
- insert the number '7' into the array while ensuring the natural order of the elements are observed if it is not?

Choose at least one correct answer

(A) Sequential Search and Straight Insertion Sort

(B) Binary Search and Binary Insertion Sort

(C) Quick Sort and Binary Search

(D) Bubble Sort and Sequential Search

(E) Merge Sort and Sequential Search

(F) Selection Sort and Binary Search

(G) Depth-first Search and Binary Insertion Sort

(H) Breadth-first Search and Bubble Sort

I Sequential Search and Bubble Sort

Question 3

2 points

Consider the following Java method which recursively finds the maximum value in an array of integers:

```
public static int findMax(int[] arr, int index) {
  if(index == 0) {
    return arr[0];
  }
  return Math.max(arr[index], findMax(arr, index - 1));
}
```

This method takes two arguments: an integer array `arr` of length n and an integer index which initially is equal to n-1 (i.e. the last index of the array).
Which **Big-O** expression correctly describes the time complexity of the method `findMax`?

Choose at least one correct answer

A O(n)

B O(1)

C O(log n)

D $O(n^2)$

E $O(2^n)$

Question 4

4 points

Imagine a scenario where an application is required to facilitate look up of the biometric data of a specified individual. The application needs to efficiently manage the association between a person's passport number and their respective biometric data. Additionally, the application must be able to handle situations where biometric data is updated or re-registered.
Considering the above requirements, which Java data structure would be the most appropriate to represent and manage the required associations?

Choose at least one correct answer

A java.util.HashSet

B java.util.LinkedList

(C) java.util.HashMap

(D) java.util.Stack

(E) java.util.TreeSet

(F) java.util.PriorityQueue

(G) java.util.ArrayList

(H) java.util.concurrent.LinkedBlockingQueue

Question 5

3 points

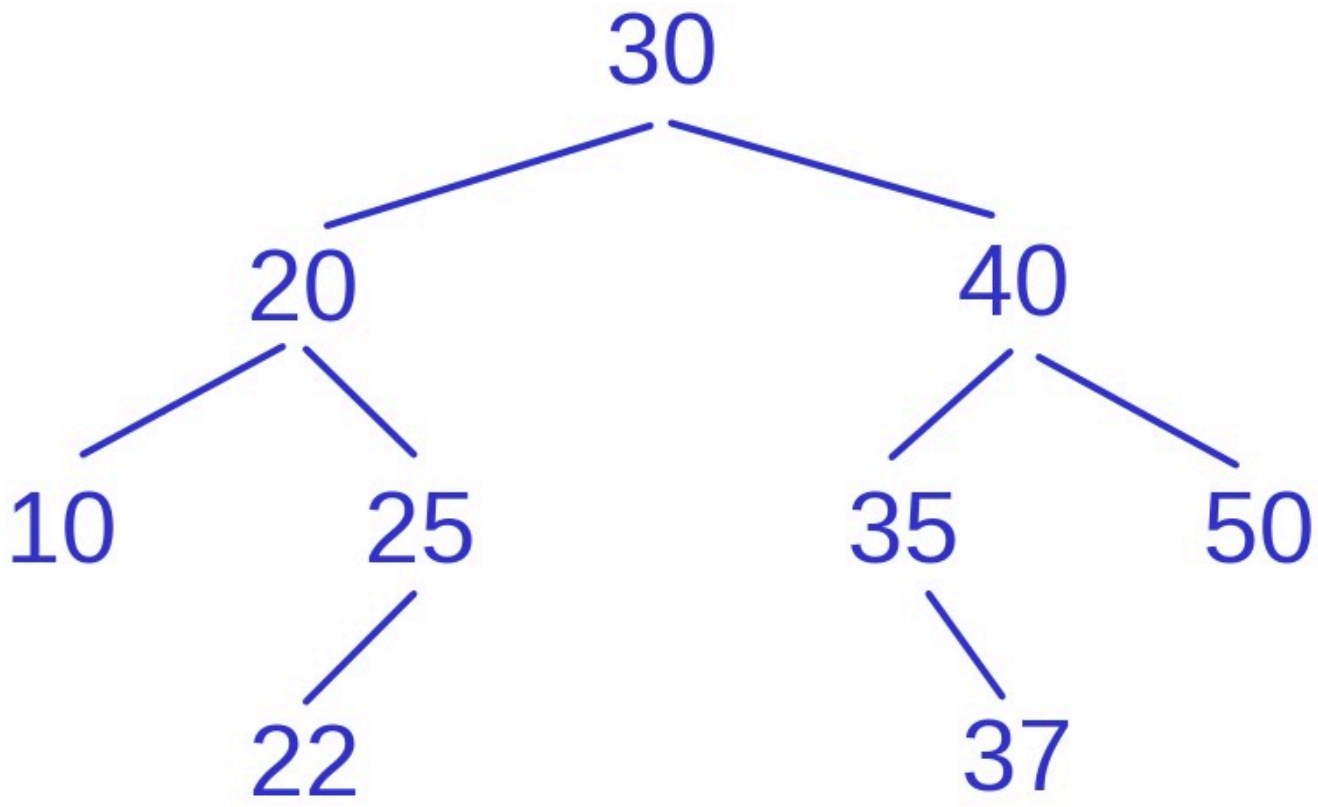
State the **asymptotic time complexity** of the algorithms described by EACH of the following growth functions:

Prompt	Answer
(1) $T(n) = n^2 \cdot (5n + \log_2(n) + 6^3)$	$O(\log n)$
(2) $T(n) = 5000$	$O(5^n)$
(3) $T(n) = 0.003n^2 + 35 + 78n \cdot \log(n)$	$O(n^3)$
	$O(1)$
	$O(n)$
	$O(n^2)$
	$O(n \log n)$
	$O(2^n)$

Question 6

6 points

Consider the following Binary Search Tree (BST):



Conducting pre-order traversal on the above BST will result in the following sequence of numbers: [Blank 1], [Blank 2], [Blank 3], [Blank 4], [Blank 5], [Blank 6], [Blank 7], [Blank 8], 50

Blank 1

Blank 2

Blank 3

Blank 4

Blank 5

Blank 6

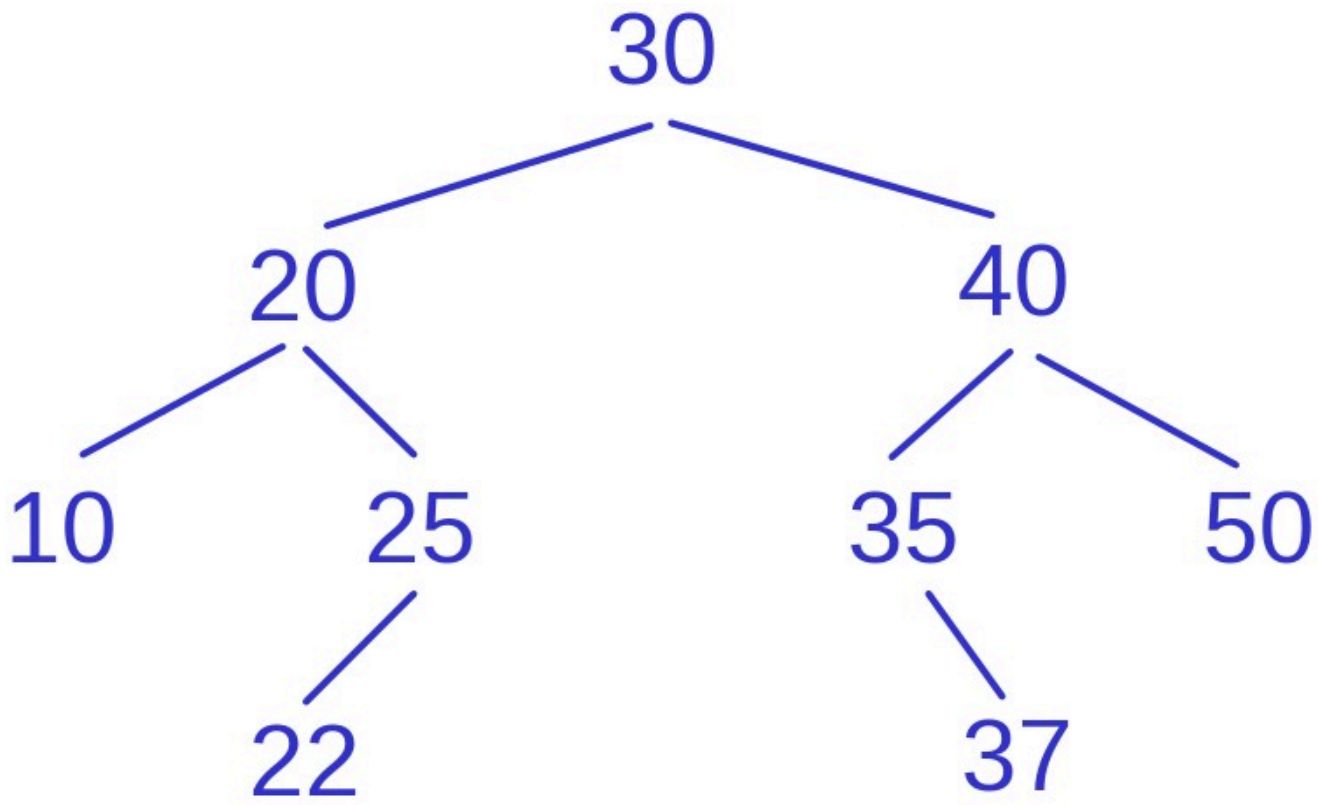
Blank 7

Blank 8

Question 7

2 points

Consider the following Binary Search Tree (BST):



When printing the elements in the above BST using in-order traversal, the third and the last elements to be printed will be the number [Blank 1] and the number [Blank 2], respectively.

Blank 1

Blank 2

Question 8

8 points

In the Java code snippet below, you are tasked with iterating through a list of colour names. The goal is to print the first character of each colour name that does not have a length of four characters and to remove from the list any colour with a name that is exactly four characters long. After these operations, the program should print the modified list.
Fill in the blanks to complete the code:

```
import java.util.*;
public class ComplexIterator {
  public static void printColor() {
    List<String> colors = new [Blank 1]<>(Arrays.asList("Red", "Green", "Blue", "Yellow", "Purple"));
    [Blank 2]<String> iter = colors.listIterator();
    while (iter.hasNext()) {
      String currentColor = iter.[Blank 3]();
      if (currentColor.length() == 4) {
        iter.[Blank 4]();
      } else {
        System.out.print(currentColor.charAt(0) + " ");
      }
```

```
    }
    System.out.println("\nModified list: " + colors);
  }
}
```

Ensure you use the fully qualified class names where necessary.

Blank 1

Blank 2

Blank 3

Blank 4

Question 9

2 points

This question is about **Big-O** notation and time complexity.
Making use of the given phrases, state the meaning of EACH of the following **Big-O** expressions:

Prompt	Answer
1 O(1)	executes in logarithmic time
2 $O(n^3)$	executes in cubic time
	executes in linear time
	executes in log-linear time
	executes in quadratic time
	executes in constant time
	executes in exponential time

Question 10

5 points

For EACH of the following statements regarding **hash tables**, state whether it is TRUE or FALSE.

Prompt	Answer

1	The size of a hash table is known as the **load factor** of the hash table.	TRUE
2	**Collision** refers to the situation when two or more elements map to the same location in a hash table.	FALSE
3	When the load factor of a hash table becomes higher than about 0.75, it is recommended to increase the size of the table in order to maintain a good performance in data access and update operations.	
4	Suppose objects `o1` and `o2` are meant to be elements stored in a hash table. If `o1.equals(o2)` returns `true`, `o1.hashCode()` may return a value that is **different** from the value returned by `o2.hashCode()` without violating any integrity constraint.	
5	A hash function uses the **division** method to compute a hash value of a given element tend to take into account of the size of the hash table in its computation.	

Question 11

4 points

Making use of the type variable `E`, fill in the blank in the following header line for a generic Java interface `IndexedListADT` defined for modelling an **indexed list**. Instances of classes which implement this interface must also support the operation of an enhanced `for` statement.

```
public [Blank 1] IndexedListADT[Blank 2] [Blank 3] [Blank 4] {
   // details omitted
}
```

Blank 1

Blank 2

Blank 3

Blank 4

Question 12

5 points

Briefly explain what is meant by the Abstract Data Type (ADT) **stack**.

Question 13

14 points

Write down the complete code, including appropriate comments, for a generic Java interface to model the Abstract Data Type (ADT) **bounded queue**.

Question 14

6 points

Consider the definition of class `LinkedStructure` in **Case Study 1**.
Write Java code for method `addToFront` in class `LinkedStructure`. This method adds a specified `String` element to the front of the structure.

Question 15

6 points

Consider the definition of class `LinkedStructure` in **Case Study 1**.
Write Java code for method `getElementAt` in class `LinkedStructure`. This method returns the `String` element at the required location within the linked structure. Position 1 refers to the first element in the structure, position 2 refers to the second element in the structure, and so on. If the specified position does not exist in the structure, a `NoSuchElementException` is thrown.

Question 16

6 points

Consider the scenario described in **Case Study 2**.
[Blank 1] is the most suitable data structure for modelling the collection of `Position` objects in class `Maze`.
To guide each ghost roaming within the maze using a trial-and-error algorithm which utilises [Blank 2] search, the Abstract Data Type (ADT) [Blank 3] may be used to store `Position` objects as the ghost moves within the maze.

Blank 1

Blank 2

Blank 3

Question 17

9 points

Consider the scenario described in **Case Study 2**.

Write a Java class to model a position and its occupant(s) in the maze. Your Java class must include declaration of the required fields and a suitable constructor for initialising each field.
Do NOT include any other method definition in your answer.
In your answer, clearly state any assumption which you have made in your class design.