

CMPT 225 D200, Fall 2026

Midterm Exam October 27, 2026

Name _____

SFU ID: |_|_|_|_|_|_|_|_|_|_|

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. The duration of the exam is 100 minutes.
2. Write your full name and SFU ID ****clearly****.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of four (4) problems, each worth 25 points
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. Explain all your answers.
8. Really, explain all your answers.

Good luck!

Problem 1 (Java syntax, Big-O notation) [25 points]

Consider the following class. Assume it has a proper constructor and all setters/getters.

```
class Book {
    String title;
    String author;
    int year;
    // Constructors, setters, getters -- assume the strings are not null
    @Override
    public boolean equals(Object other) {
        if (this==other)
            return true;
        if (other==null || !(other instanceof Book))
            return false;

        Book book = (Book) other;
        return year == book.year &&
            title.equals(book.title) &&
            author.equals(book.author);
    }
}
```

A. (10 points) What will be the output of the following code? Explain

```
Book b1 = new Book("Dune", "Frank Herbert", 1965);
Book b2 = new Book("Dune", "Frank Herbert", 1965);
Book b3 = b1;

if (b1==b3)
    System.out.println("b1==b3 is true");
if (b2==b3)
    System.out.println("b2==b3 is true");
if (b2.equals(b3))
    System.out.println("b2.equals(b3) is true");
```

B. (5 points each) Apply the Master Theorem to the following recursive definition, and express the running time as a closed formula.

- $T(n) = 2 T(n/2) + \Theta(n^2)$, $T(1) = O(1)$

- $T(n) = 4 T(n/2) + \Theta(n)$, $T(1) = O(1)$

C. (5 points) Write a function that gets a parameter n , and its running time on the input n can be expressed as $T(n) = 2T(n/2) + O(n)$ with $T(1) = O(1)$.

```
void foo(int n) {
```

```
}
```

Problem 2 (Linked Lists) [25 points]

Implement a class `LinkedList` that supports the following operations with running time $O(1)$ for each operation.

```
public class LinkedList<T> {  
    public LinkedList() // constructor creates an empty list  
    public void addFirst(T item) // adds item to the front of the list  
    public T getFirst() // returns the first item in the list  
    public T removeFirst() // removes the first item and returns it  
    public void addLast(T item) // adds item to the end of the list  
    public T getLast() // returns the last item in the list  
    public T removeLast() // removes the last item and returns it  
    public int size() // returns the number of elements in the list  
    public boolean isEmpty() // checks if the list is empty  
}
```

Before writing code, explain your answer.



Problem 3 (Binary Trees) [25 points]

In this problem use the following definition of Binary Tree. You may assume the classes have the standard getters/setters.

```
public class BTreeNode<T> {  
    private T data;  
    private BTreeNode<T> leftChild;  
    private BTreeNode<T> rightChild;  
    private BTreeNode<T> parent;  
}  
  
public class BinaryTree<T> {  
    private BTreeNode<T> root;  
}
```

- A. (7 points) Draw a binary tree whose inOrder sequence is [1,2,3,4,5,6,7,8,9] and preOrder is [6,2,1,4,3,5,7,8,9].

- B. (18 points) Implement the class `PreOrderIterator` for the class `BinaryTree`. Your class needs to implement the following methods:

```
public PreOrderIterator(BinaryTree<T> tree) // a constructor
public boolean hasNext() // checks if the traversal has more vertices
public T next() // returns the next item in the pre order
```

All operations must run in $O(1)$ time in the worst-case.

Explain the idea first. Specify all private data fields and the algorithm.

```
public class PreOrderIterator<T> implements Iterator<T> {
```

```
}
```

Problem 4 (Binary Search Trees) [25 points]

A. (5 points) Consider the class `BinarySearchTree`, defined as follows.

```
public class BinarySearchTree<T extends Comparable<T>> {  
    private BTNode<T> root;  
};
```

Use the definition of `BTNode<T>` from Problem 3.

Write the method `max()` for the class `BinarySearchTree` that returns the maximal element in the tree. If the tree is empty, the method returns `null`.

```
public T max() {
```

```
}
```

B. (5 points) Start with an empty tree

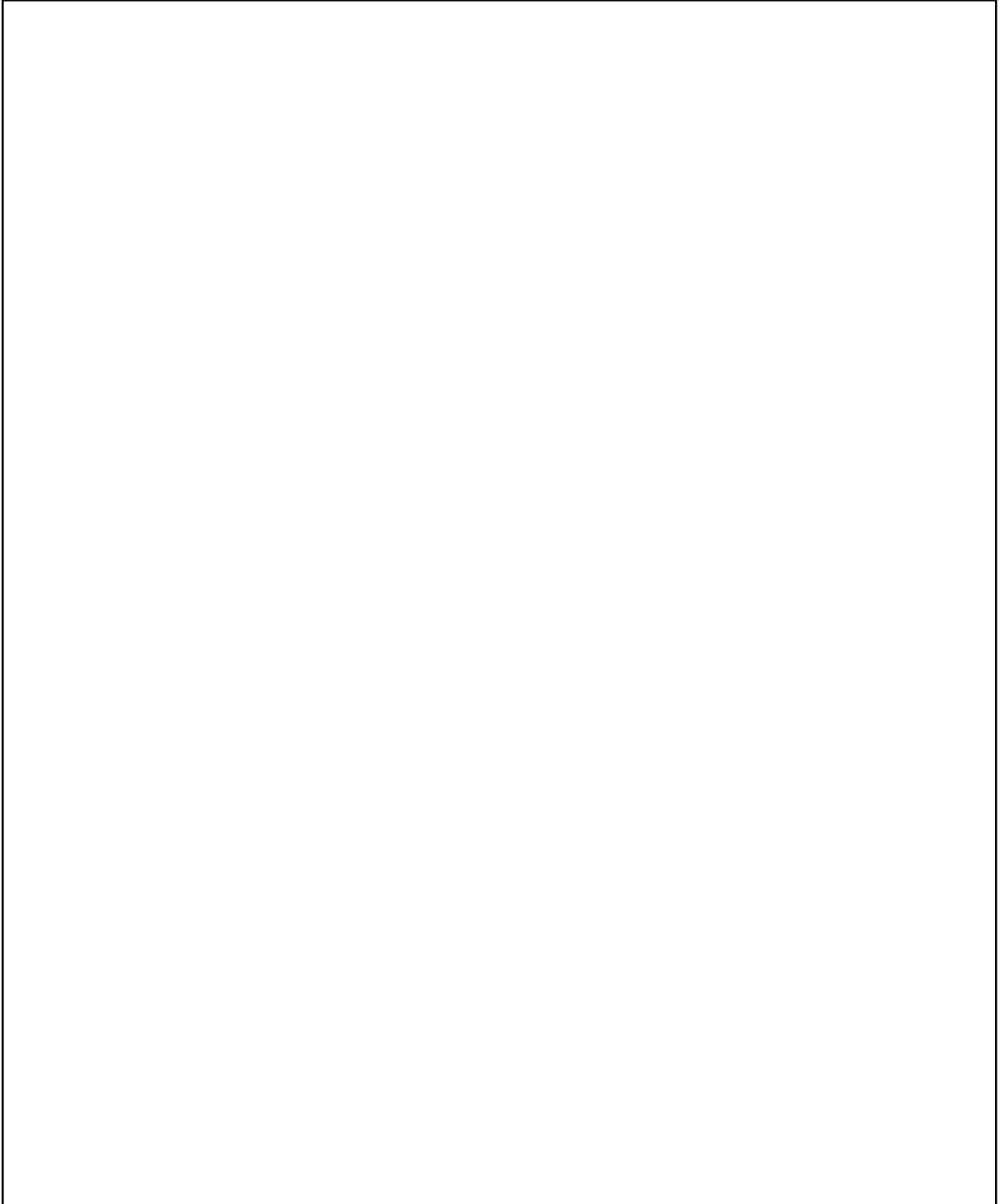
- Insert the elements [5, 1, 4, 7, 6, 8, 2, 3], and draw the obtained tree
- Remove 4 and draw the obtained tree
- Remove 5 and draw the obtained tree

- C. (15 points) Write a function that gets an array with the post order traversal of a Binary Search Tree, and reconstructs the tree. The function returns the root of the tree. Use the definition of `BTNode<T>` from Problem 3.

Explain your answer.

```
public static <T extends Comparable<T>>  
    BTNode<T> postOrder2BST(T[] postOrder)
```

Extra page



Master Theorem

Let $T(n) = a T(n/b) + f(n)$, $T(1) = O(1)$

Define $c = \log_b(a)$

- If $f(n) = \Theta(n^d)$ for $d < c$, then $T(n) = \Theta(n^c)$
- If $f(n) = \Theta(n^d)$ for $d > c$, then $T(n) = \Theta(n^d)$
- If $f(n) = \Theta(n^c)$, then $T(n) = \Theta(n^c \log(n))$