048

CSC 115

Midterm Exam:

Thursday, July 28, 2022

Exam duration: 75 minutes

Instructor: Celina Berg

Name: Arfaz Hossain (please print clearly!)

UVic ID number: V00984826 (please print clearly!)

Signature: Arfaz Hussain

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.

- We will not answer questions during the exam. If you feel there is an error or ambiguity, write your assumption and answer the question based on that assumption.
- Answer all questions on this exam paper.
- The exam is closed book. No books or notes are permitted.
- **Electronic devices, including calculators, are not permitted.**
- The marks assigned to each part are printed within brackets. Partial marks are available.
- There are fourteen (16) pages in this document, including this cover page.
- Pages 6, 10 and 16 is left blank for scratch work. If you write an answer on that page, clearly indicate this for the grader under the corresponding question.
- Clearly indicate only one answer to be graded. Questions with more than one answer will be given a **zero grade.**
- It is strongly recommended that you read the entire exam through from beginning to end before beginning to answer the questions.

04

Part 1 (38 marks)

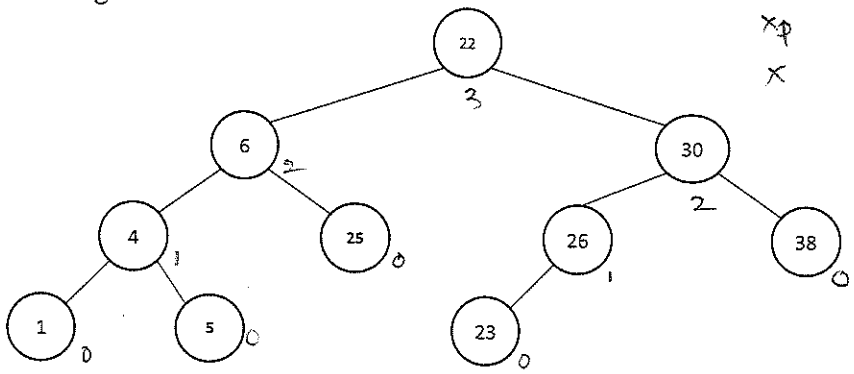
For the following questions, write your final answer in the box provided.

a) What is the output of the following program?

```
public class Foo {
    public static void main(String[] args) {
        ~~System.out.print("A ")~~;
        Bar b = new Bar(10);
        try {
            ~~System.out.print("B ");~~
            b.fn(15);
            System.out.print("C ");
        } catch (BException e) {
            System.out.print("D ");
        }
        System.out.print("E ");
    }
}

public class Bar {
    private int x;
    public Bar(int x) {
        this.x = x;    → 10
    }                → 15

    public int fn(int z) throws BException {
        try {
            System.out.print("F ");
            Baz.fn(z,x);        15, 10
        } catch (AException e) {
            System.out.print("G ");
        }
        if (x>0) {
            System.out.print("H ");
            throw new BException();
        }
        System.out.print("I ");
        return x+1;
    }
}
                          ↙ 15  ↗ 10
public class Baz {
    public static int fn(int a, int b) throws AException {
        System.out.print("J ");
        if (a<b) {
            throw new AException();
        }
        System.out.print("K ");
        return a+b;
    }          25
}

public class BException extends Exception {
}

public class AException extends Exception {
}
```

Final answer: A B F J K H D E

b) Consider the following tree:



×φ
×

Add an **X** to the box beside all terms that correctly describe this tree. (negative marks for incorrect selections)

× ☐ Complete

× ☐ Full (or perfect)

× ☑ Binary Search Tree

√ ☑ Balanced

× ☐ Max Heap

√ ☑ Binary Tree

	1	0
c	2i 2i+1	2i+1 2i+2
p	i/2	$\frac{i-1}{2}$
	0	

q:

2(32) = 64 +1
= 65

42/2 = 21

0 - 1 2 3

c) You are given an array-based implementation of a binary tree and you are told the root is at index 1 of the array. Answer the following with this in mind:

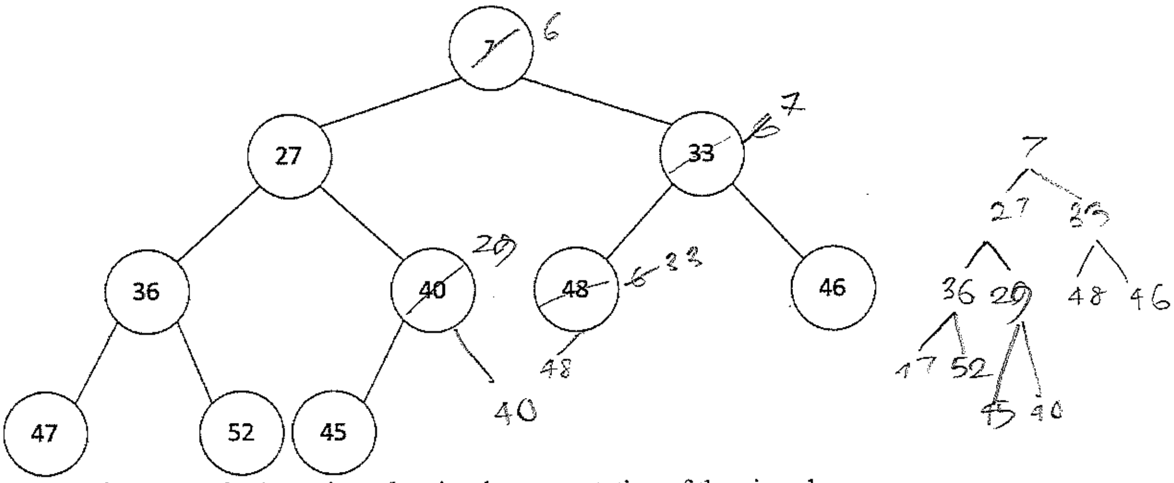
Consider the entry at index 25 of the array, what is the index of that entry's left child? 50

Consider the entry at index 32 of the array, what is the index of that entry's right child? 65

Consider the entry at index 43 of the array, what is the index of that entry's parent? 21

d) For the following questions consider the representation below of a min heap (smallest value in the tree is stored at the root). Assume this heap has an array-based implementation with the fields:

```
int[] data;    // an array to hold values in the heap
int size;      // the number of elements in the heap
public static final int root = 1; // index of the root in data
```



i. Fill in the state of `data` and `size` given the visual representation of the given heap.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	-	7	27	33	36	40	48	46	47	52	45	-	-	-	-

size	10

ii. The values 29 and 6 are to be inserted into the given heap.
Fill in the state of `data` and `size` **after the 29** has been inserted:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	-	7	27	33	36	29	48	46	47	52	45	40	-	-	-

size	11

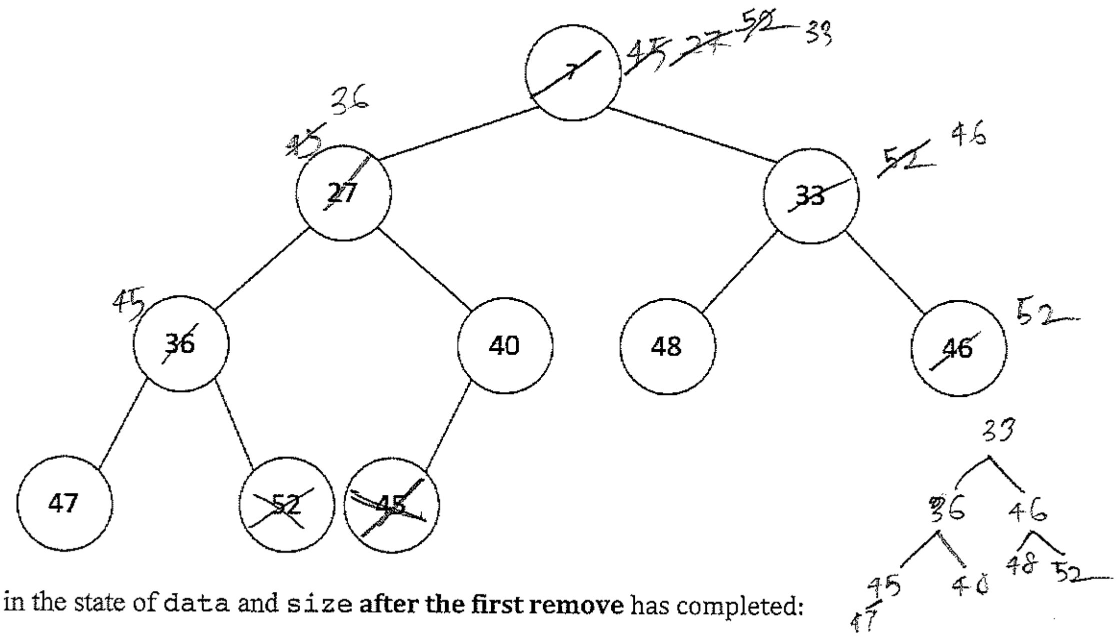
Fill in the state of `data` and `size` **after both the values 29 and 6** have been inserted:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	-	6	27	7	36	29	33	46	47	52	45	40	48	-	-

size	12

iii. Assume the two smallest values are removed from the given min heap.
NOTE: Use the original given heap, NOT the heap produced after the insert in part ii of this question. A copy of this min heap is replicated here for your convenience:

```
int[] data;    // an array to hold values in the heap
int size;      // the number of elements in the heap
public static final int root = 1; // index of the root in data
```



Fill in the state of data and size **after the first remove** has completed:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	–	27	36	33	45	40	48	46	47	52	–	–	–	–	–

size

9

Fill in the state of data and size **after the first and second remove** have completed:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data	–	33	36	46	45	40	48	52	47	–	–	–	–	–	–

size

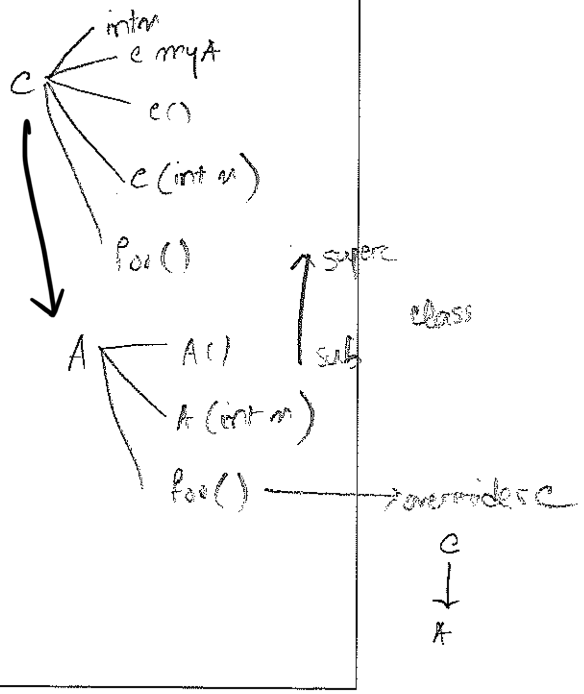
8

0.

Page left blank intentionally for scratch work if needed...
If you write an answer on this page, clearly indicate this for the grader under the corresponding question.

e) Given the following classes:

```
public class A extends C {
    public A() {
        System.out.println("constructor A 1");
    }
    public A(int x) {
        System.out.println("constructor A 2");
    }
    public void foo() {
        System.out.println("foo A:" + x);
    }
}
public class C {
    public int x;
    public C myA;

    public C() {
        System.out.println("constructor C 1");
        this.x = 0;
    }
    public C(int x) {
        this.x = x;
        myA = new A(x-2);
        System.out.println("constructor C 2");
    }
    public void foo() {
        System.out.println("foo C:" + myA.x);
    }
}
```



What is the output if the following code snippet is executed:

```
C myC = new C(5);
myC.foo();
```

constructor C1

✓ constructor A 2

constructor C 2

✓ ~~foo C :5~~

foo C:0

What is the output if the follow code snippet is executed:

```
C myC = new A(5);
myC.foo();
```

constructor C1

constructor A 2

foo A : 5

0.

f) Given the following classes:

```
class A {
    public void doSomething() {
        System.out.println("The sun is shining");
    }
}
class B extends A {
    public void doSomething() {
        System.out.println("It is raining again");
    }
}
class C extends B {
    public void doSomething(String s) {
        System.out.println("repeat" + s);
    }
}
```

A
↓
B
↓
C

✓ i. What is the output if the following code is compiled and run? (write answer in box provided)

```
A a = new A();
a.doSomething();
```

The sun is shining

✓ ii. What is the output if the following code is compiled and run? (write answer in box provided)

```
B b = new B();
b.doSomething();
```

It is raining again

✓ iii. What is the output if the following code is compiled and run? (write answer in box provided)

```
C c = new C();
c.doSomething();
```

It is raining again

✓ iv. What is the output if the following code is compiled and run? (write answer in box provided)

```
A a = new A();
C c = new C();
a = (A) c;
a.doSomething();
```

It is raining again

✗ v. What is the output if the following code is compiled and run? (write answer in box provided)

```
A a = new A();
B b = new B();
b = (B) a;
b.doSomething();
```

Class Cast Exception.

g) Given the hash table below that uses separate chaining, insert the elements:
1, 8, 10, 99, 13
Assume the hash function is h(*k*) = *k* % 7

0	1	2	3	4	5	6
–	1 ↓ 8 ↓ 99	–	10	–	-	13

1%7= 1
8%7 = 1
10%7 = 3
99%7 = 1
13%7 = 6

h) Given the hash table below that uses open addressing with linear probing, insert the elements:
1, 8, 10, 99, 13
Assume the hash function is h(*k*) = *k* % 7

0	1	2	3	4	5	6
–	1	8	10	99	–	13

i) What is the output of the following code snippet when compiled and run:

```
List<Integer> numbers = new LinkedList<Integer>();
for (int i=1; i<=10; i++) {
    numbers.add(i);
}
Iterator<Integer> iter = numbers.iterator();
while(iter.hasNext()) {
    System.out.print(iter.next() + " ");
    iter.next();
}
```

1
2
3
4
5
6
7
8
9
10

1 3 5 7 9

j) What is the output of the following code snippet when compiled and run:

```
List<Integer> numbers = new LinkedList<Integer>();
for (int i=1; i<=5; i++) {
    numbers.add(i);
}
Iterator<Integer> iter = numbers.iterator();
while(iter.hasNext()) {
    System.out.print(iter.next() + " ");
    iter.next();
}
```

1
2
3
4
5
~~6~~
~~7~~

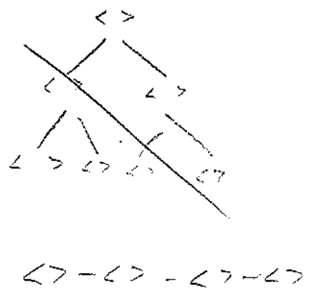
1 3 5 ↑ NULLPOINTER EXP

Page left blank intentionally for scratch work if needed…
If you write an answer on this page, clearly indicate this for the grader under the corresponding question.

k) Assume you have implemented a `LinkedList` class that holds `key:value` pairs where `keys` are unique in the list and all of the methods work correctly. Note, the implementation does not guarantee the pairs are in any particular order.

You decide to extend the `LinkedList` class with a `SortedLinkedList` class in which the pairs are in increasing sorted order by `key`.

Add an **X** to the box beside the methods you would override in the `SortedLinkedList` class. (negative marks for incorrect selections)



☒ print the elements in the list from front to back

✓ X ☒ given a `key` and a `value`, inserts a the given `key:value` pair into the list

X □ prints the elements in the list in increasing sorted order by `key`

X □ given a `key`, finds and removes the corresponding `key:value` pair from the list

☒ given a valid position in the list, returns the `key:value` pair at that position

□ returns the number of elements in the list

X □ given a `key`, finds the `key:value` pair from the list with that `key` and returns the corresponding `value`

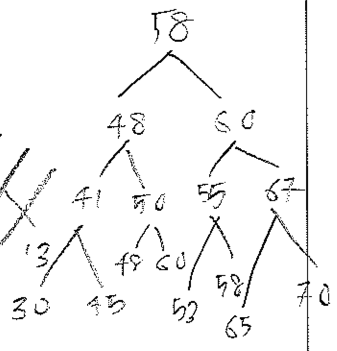
☒ given a `value`, finds the first `key:value` pair from the list with that `value` and returns the corresponding `key`

Part 2 (10 marks)

Consider the implementation of a Binary Search Tree ADT that holds elements of type `int`.
You are to implement the sumAbove method according to the given documentation.

(curr == null)

```
public class TreeNode {
    protected int value;
    protected TreeNode left;
    protected TreeNode right;

    public TreeNode(int value) {
        this.value = value;
        this.left = null;
        this.right = null;
    }

    public int getValue() {
        return this.value;
    }

    public void setValue(int value) {
        this.value = value;
    }

    public TreeNode getLeft() {
        return this.left;
    }

    public void setLeft(TreeNode left) {
        this.left = left;
    }

    public TreeNode getRight() {
        return this.right;
    }

    public void setRight(TreeNode right) {
        this.right = right;
    }
}// END of TreeNode Class


public class BinarySearchTree {
    private TreeNode root;

    public BinarySearchTree() {
        this.root = null;
    }

    /* Method Name: sumAbove
     *
     * Purpose: returns the sum of all values in this BinarySearchTree
     * that are above the given threshold
     *
     * Parameters: int threshold
     *
     * Returns:  int – the sum
     */
     // method implementation to be completed on the following page
```



```
public int sumAbove (int threshold) {
    if ( this.root == null) { return 0;}
    else {
        return sumAboveRec ('this root, threshold, 0);
    }
public int sumAboveRec(TreeNode curr, int t, int sum){
    if (curr==null) { return sum;}
    else if (curr.getValue().compareTo(t)>0) {
            sum += curr.getValue();
            return (sum + sumAboveRec (curr.getRight(), t, sum)
                          + sumAboveRec( curr getLeft(), t, sum));

    } else {
            sum+=0;
            return (sumAboveRec(curr.getRight (), t, sum) +
                          sumAboveRec( curr.getLeft(), t, sum) +
                                                          sum);

    }
}
```

}// end of BinarySearchTree class

6

Part 3 (12 marks)

Consider following Map ADT interface and implementation and the Song and MyTunes classes. You are to use these in your implementation of the addSong method in the MyTunes class.

```
public class KeyNotFoundException extends RuntimeException {
   // uses Java's default constructor
   // Reminder: RuntimeException is an unchecked Exception type
} // END of KeyNotFoundException

public interface Map<K extends Comparable<K>, V> {
    /* Purpose: Returns true if key is in this Map, false otherwise
     * Parameters: K key
     * Returns: boolean
     */
    boolean containsKey(K key);

    /* Purpose: Returns the value associated with given key in this Map
     * Parameters: K key
     * Returns: boolean
     * Throws: KeyNotFoundException Runtime exception if key is not found
     */
    V get (K key) throws KeyNotFoundException;

    /* Purpose: If the given key is not in this Map,
     *   a new entry with given key and value is inserted into this Map.
     *   Otherwise the existing value associated with the existing key
     *   is replaced with the given value.
     * Parameters: K key
     * Returns: nothing
     */
    void put (K key, V value);

} // END of Map Interface


public class MapImpl<K extends Comparable<K>, V> implements Map<K, V> {
   // assume this class has been implemented for you - DO NOT complete it
} // END of MapImpl class


public class Song {
    private String  title;
    private String  artist;
    private int     length; // in seconds

    public String getArtist() {
        return artist;
    }
} // END of Song class
```

s < title
artist
int length

```
public class MyTunes{
    private Map<String, List<Song>> songs;

    public MyTunes() {
        songs = new MapImpl<String, List<Song>>();
    }
    /* Method Name: addSong
     * Purpose: Adds given Song s to songs Map where,
     *         the artist of s is the key.
     *  If the artist of s is not a key in songs, a new entry is created,
     *  otherwise Song s is added to the value list of the existing entry.
     * Parameters: Song s
     * Returns: nothing
     */
    // method implementation to be completed here:

      public void addSong (Song s) {

         Iterator<List <Song>> newList = MyTunes.Iterator();

         while (newList.hasNext()) {
               if (newList.containsKey.compareTo( s.getArtist( ) ) ) ) {

         }

}// end of MyTunes class
```

Page left blank intentionally for scratch work if needed...
If you write an answer on this page, clearly indicate this for the grader under the corresponding question.

END OF THE EXAM