



King Saud University

College of Computer and Information Sciences

Department of Computer Science

Data Structures CSC 212

Midterm Exam Solution - Spring 2019

Date: _____

Duration: 90 minutes

Guidelines

- No calculators or any other electronic devices are allowed in this exam.

Student ID:

Name:

Section:

Instructor:

1	2	3.1	3.2	Total

Question 1 30 points

- Trace the execution of the following expression: **8 6 4 - / 6 4 3 % 2 + - > 8 7 2 1 - + ≠ &&**.

Show the content of the data structure(s) **after** parsing each operation.

-	/	%	+	-
<div>2</div> <div>8</div>	<div>4</div>	<div>1</div> <div>6</div> <div>4</div>	<div>3</div> <div>6</div> <div>4</div>	<div>3</div> <div>4</div>
>	-	+	≠	&&
<div>T</div>	<div>1</div> <div>7</div> <div>8</div> <div>T</div>	<div>8</div> <div>8</div> <div>T</div>	<div>F</div> <div>T</div>	<div>F</div>

2. Trace the execution of the following expression: $((4 + 2) / (7 \% 4)) + 2 - (5 * 2 + (6 / 3)) + 2$.

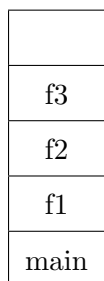
Draw the content of the data structure(s) after parsing each operation.

+	/	%	+	-
<div> <div>+</div> <div>(</div> <div>4 (</div> </div>	<div> <div>/</div> <div>6 (</div> </div>	<div> <div>%</div> <div>(</div> <div>7 /</div> <div>6 (</div> </div>	<div> <div>2 +</div> </div>	<div> <div>4 -</div> </div>
*	+	/	+	\$
<div> <div>*</div> <div>5 (</div> <div>4 -</div> </div>	<div> <div>+</div> <div>10 (</div> <div>4 -</div> </div>	<div> <div>/</div> <div>(</div> <div>6 +</div> <div>10 (</div> <div>4 -</div> </div>	<div> <div>-8 +</div> </div>	<div> <div>-6 \$</div> </div>

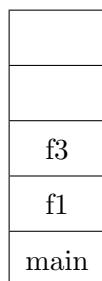
3. Consider the following code:

```
public static void f1(int n) {
    n++;
    f3(n);
    f2(n);
    f3(n); }
public static void f2(int n) {
    f3(n);
    n++;
    f3(n); }
public static void f3(int n) {
    System.out.println(n); }
public static void main(String[] args) {
    int n = 3;
    f2(n);
    n++;
    f1(n); }
```

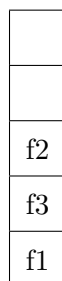
For each snapshot of the call stack below, indicate if it is valid for this code.



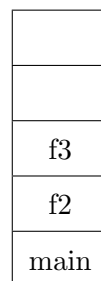
(A) Yes (B) No



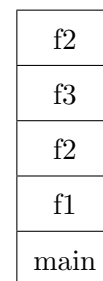
(A) Yes (B) No



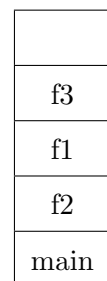
(A) Yes (B) No



(A) Yes (B) No



(A) Yes (B) No



(A) Yes (B) No

Question 2 15 points

- To solve the problem, we can use a data structure similar to a queue, where in addition to the data, we keep track of the frequency of each element. We will call this ADT: FreqQueue. It has the following

specification:

Domain:

- Data: generic, T.
- Structure: linear.

Operations: All operations are done on a FreqQueue fq.

- – procedure length(int: n)
 - Requires: none.
 - Result: Returns the number of elements in fq.
 - Output: n.
- – procedure full(boolean: f)
 - Requires: none.
 - Result: Returns true if fq is full, false otherwise.
 - Output: f.
- – procedure add(T: e)
 - Requires: fq is not full.
 - Result: If e does not exist in fq, then it is added at the end of fq with frequency 1, otherwise the frequency of e is incremented by 1.
 - Output: none.
- – procedure serve(T: e, int: f)
 - Requires: fq is not empty.
 - Result: the first element in fq is removed, its value is assigned to e, and its frequency is assigned to f.
 - Output: e, f.

2.

```
public interface FreqQueue<T> {
    int length();

    boolean full();

    void add(T e);

    FreqQueueElem<T> serve();
}
```

The return type of the method serve is :

```
public class FreqQueueElem<T> {
    public T data;
    public int freq;

    public FreqQueueElem(T data, int freq) {
        this.data = data;
        this.freq = freq;
    }
}
```

Question 3..... 55 points

1.

```
import java.util.Random;

public class ArrayRandomSet<T> implements RandomSet<T> {
    private T[] data;
    private int n, maxSize;
    private Random rnd;

    public ArrayRandomSet(int maxSize) {
        this.maxSize = maxSize;
        n = 0;
        data = (T[]) new Object[maxSize];
        rnd = new Random();
    }

    @Override
    public int size() {
        return n;
    }

    @Override
    public boolean full() {
        return n == maxSize;
    }

    @Override
    public boolean add(T e) {
        for (int i = 0; i < n; i++) {
            if (e.equals(data[i])) {
                return false;
            }
        }
        data[n] = e;
        n++;
        return true;
    }

    @Override
    public T remRandom() {
        int k = rnd.nextInt(n);
        T e = data[k];
        data[k] = data[n - 1];
        n--;
        return e;
    }
}
```

2.

```
public static <T> boolean find(RandomSet<T> s, T e) {
    int n = s.size();
    RandomSet<T> ts = new ArrayRandomSet<T>(n);
    boolean found = false;
    for (int i = 0; i < n; i++) {
        T tmp = s.remRandom();
        ts.add(tmp);
        if (e.equals(tmp)) {
            found = true;
            break;
        }
    }

    while (ts.size() > 0) {
        s.add(ts.remRandom());
    }
    return found;
}
```