CSC 1351: Exam 3

VERSION A

Name: _____

1 Files and Exceptions

1.1

Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class Stacker {
  static void stack(int n) {
    if(n == 0) throw new RuntimeException();
    stack(n-1);
  }
  public static void main(String[] args) {
    Integer a = null;
    try {
      stack(4);
    } catch(RuntimeException re) {
      re.printStackTrace();
      ..........
    }
  }
}
```

// output:

```
java.lang.RuntimeException
        at Stacker.stack(Stacker.java:3)
        at Stacker.stack(Stacker.java:4)
        at Stacker.stack(Stacker.java:4)
        at Stacker.stack(Stacker.java:4)
        at Stacker.stack(Stacker.java:4)
        at Stacker.main(Stacker.java:9)
```

1.2

What is the output generated by the following code?

```
class Exa extends RuntimeException {}
class Exb extends RuntimeException {}
public class Flow {
  public static void main(String[] args) {
    try {
    try {
      System.out.println("step␣1");
      if(true) throw new Exa();
    } catch(Exa a) {
      System.out.println("step␣2");
      if(true) throw new Exb();
    } finally {
      System.out.println("step␣3");
    }
    System.out.println("step␣4");
```

```
    } catch(Exception ex) {}
  } }
```

// output?

```
step 1
step 2
step 3
```

1.3

Fill in the missing code.

```
    String in = "Mon␣Jan␣2,␣1982\nWed␣Aug␣14,␣1979\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {
                      "(\\w+)\\s+(\\w+)\\s+(\\d+),\\s*(\\d+)";
      String pattern = ----------------------------------------
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();
                         mr.group(1)
        String weekday = -----------;
                         mr.group(2)
        String month = -----------;
                   Integer.parseInt(mr.group(3))
        int day = ----------------------------;
                    Integer.parseInt(mr.group(4))
        int year = ----------------------------;
        System.out.printf("Date:␣%s␣%d/%s/%d%n",weekday,day,month,year);
      }  else { System.out.println("Incorrect");
      } s.nextLine();
```

// output:

```
Date: Mon 2/Jan/1982
Date: Wed 14/Aug/1979
```

1.4

Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.io.File;
--------------------
import java.io.IOException;
import java.util.Scanner;
public class FilesRead {
  public static void main(String[] args) throws IOException {
    String fileName = "/etc/group";
    File file = new File("/etc/group");
    --------------------
    if(file.exists()) {
      Scanner s = new Scanner(file);
      while(s.hasNextLine())
        System.out.println(s.nextLine());
    } } }
```

2 Other

2.1

Fill in the missing code.

```
// interface Runnable {
//   public void run();
// }
public class Anon {
  public static void main(String[] args) {
                  new Runnable() {
    Runnable r = ----------------
      public void run() {
      --------------------
        System.out.println("Hello, world");
        ---------------------
      }
      --------------------
    };
    r.run();
  }
}
```

// output:

```
Hello, world
```

3 Linked Lists

4 Stacks and Queues

4.1

What is the difference between a Java `Set` and a `List`?
Elements in a `Set` have to be unique, while they don't need to be for a `List`.

4.2

Which interface directly extends the `Collection` interface and is implemented in the `LinkedList` class?
List

4.3

Name three classes within the Java collections framework that implement the `List` interface.
ArrayList, LinkedList, Vector, (Stack)

4.4

Name two interfaces that `Stack` is implementing.
List, Collection, (Iterable)

4.5

Assume elements 2, 4, 6, 8 being put element-wise first into a queue, taken out again, then put onto a stack, taken out again, put into a queue, and taken out again. In which order do you now have these elements, and which order were they after each step?
after queue: 2, 4, 6, 8 (original)
after stack: 8, 6, 4, 2 (reverse)
after queue: 8, 6, 4, 2 (reverse)

4.6

What is wrong with the following code to evaluate the top of the operators and numbers stack of an expression evaluator? How could you fix it?

```
static Stack<Double> numbers = new Stack<>();
static Stack<String> operators = new Stack<>();
static void evaluateTop() {
    Double n2 = numbers.peek();
    Double n1 = numbers.peek();
    String op = operators.peek();
    if (op.equals("-"))
        numbers.push(n1-n2);
    else if (op.equals("+"))
        numbers.push(n1+n2);
    else if (op.equals("*"))
        numbers.push(n1*n2);
    else if (op.equals("/"))
        numbers.push(n1/n2);
}
```

The `peek()` don't remove elements off the stack, use `pop()` instead.

4.7

Complete the code within the `contains()` method, returning `true` if the doubly linked list contains an element with the value of i, and `false` otherwise.

```
class Node {
    Integer data;
    Node previous;
    Node next;
}

public class MyList {
    Node start;
    Node end;

    public boolean contains(Integer i) {
        if (start == null) return false;
        Node current = start;
        while (current != null) {
            if (i.equals(current.data)) return true;
            current = current.next;
        }
        return false;
    }
}
```