CSC 1351: Exam 3

PRACTICE VERSION

Name: _____

1 Files and Exceptions

1.1

Does the following code compile? If it does not, how can it be fixed? If it does, what is it's output? Does it throw an exception? If so, how can it be fixed?

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

1.2

Fill in the missing code.

```
public class ShowF {
  public static void main(String[] args) {
    double x = 1.23456789;
    System.out.println("Formatting:");

    System.out.print(String.format(--------,3));

    System.out.print(String.format(--------,x));
  }
}
```

// output:

```
Formatting:
003
1.23
```

1.3

Fill in the missing code.

```
public class ShowF2 {
  public static void main(String[] args) {
    double x = 98.76543;
    System.out.println("Formatting:");

    System.out.print(String.format(---------,":"));

    System.out.print(String.format(--------,x));
  }
}
```

// output:

```
Formatting:
(   :)
98.765
```

1.4

Fill in the missing code.

```
    String in = "9+␣104␣=113\n8␣+4=␣5\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {

      String pattern = ----------------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();

        int a = Integer.parseInt(-----------);

        int b = Integer.parseInt(-----------);

        int c = Integer.parseInt(-----------);
        if(a + b == c) System.out.println("Correct");
        else System.out.println("Incorrect");
      } s.nextLine();
```

// output:

```
Correct
Incorrect
```

1.5

Fill in the missing code.

```
    String in = "Methuselah,␣age:938\nWilma,age:␣45\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {

      String pattern = ------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();

        int a = Integer.parseInt(-----------);

        String b = -----------;
        System.out.printf("Name:␣%s,␣age:␣%d%n",b,a);
      } s.nextLine();
```

// output:

```
Name: Methuselah, age: 938
Name: Wilma, age: 45
```

1.6

Fill in the missing code.

```
    String in = "Deadpool␣891-22-3954\n"+
                "Wade␣113-24-5566\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {

      String pattern = --------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();

        int a = Integer.parseInt(-----------);

        int b = Integer.parseInt(-----------);

        int c = Integer.parseInt(-----------);

        String d = -----------;
        System.out.printf("Name:␣%s,␣ssn:␣%d-%d-%d%n",d,a,b,c);
      } s.nextLine();
```

// output:

```
Name: Deadpool, ssn: 891-22-3954
Name: Wade, ssn: 113-24-5566
```

2 Exceptions

2.1

It's so hard to say goodbye. What is the output generated by the following code?

```
public class Code {
  public static void main(String[] args) {
    try {
      try {
        System.out.println("Hello␣Alice");
        if(true) throw new NullPointerException("The␣End");
        System.out.println("Goodbye␣Alice");
      } finally {
        System.out.println("Hello␣Bob");
      }
      System.out.println("Goodbye␣Bob");
    } catch(NullPointerException npe) {}
  }
}
```

// output?

2.2

When is an infinte loop not infinite? What is the output generated by the following code?

```
public class Code {
  public static void main(String[] args) {
    int sum = 0, count = 0;;
    int[] arr = new int[]{1,2,3,2,1};
    try {
      int n = 0;
      while(true) {
        sum += arr[n++];
        count++;
      }
    } catch(ArrayIndexOutOfBoundsException ex) {}
    System.out.println("sum = "+sum);
  }
}
```

// output?

2.3

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

```
public class Npe {
  public static void main(String[] args) {
    Integer a = null;
    try {
      int b = a + 3;
    } catch(NullPointerException npe) {

      ..........
    }
  }
}
```

// output:

```
java.lang.NullPointerException
        at Npe.main(Npe.java:5)
```

2.4

What is the bank balance at the end of main? Why?

```
public class Bank {
  double balance;
  Bank(double b) { balance = b; }
  void debit(double amount) throws Exception {
    if(amount > balance) throw new Exception();
    balance -= amount; }
  public static void main(String[] args) throws Exception {
    Bank b = new Bank(5.2);
    try {
      b.debit(10.0);
    } catch(Exception e) {
      b.debit(1.0);
    }
  }
}
```

2.5

What is the bank balance at the end of main? Why?

```
public class Bank {
  double balance;
  Bank(double b) { balance = b; }
  void debit(double amount) throws Exception {
    if(amount > balance) throw new Exception();
    balance -= amount; }
  public static void main(String[] args) throws Exception {
    Bank b = new Bank(5.2);
    try {
      b.debit(2.0);
    } catch(Exception e) {
      b.debit(1.0);
    }
  }
}
```

3 Linked Lists

3.1

What advantages do linked lists have over arrays (2)?

3.2

What disadvantages do linked lists have over arrays (2)?

3.3

Suppose the list `letters` contains elements “S”, “P”, “R”, “I”, “N”, and “G”. Draw the contents of the list and the iterator position for the following operations:

```
ListIterator<String> iter = letters.iterator();
iter.next();
iter.remove();
iter.next();
iter.remove();
iter.next();
iter.add("A");
iter.next();
iter.next();
iter.next();
iter.remove();
```

3.4

Suppose the list `letters` contains elements “S”, “P”, “R”, “I”, “N”, and “G”. Draw the contents of the list and the iterator position for the following operations:

```
ListIterator<String> iter = letters.iterator();
iter.next();
iter.next();
iter.remove();
iter.next();
iter.remove();
iter.next();
iter.remove();
iter.add("U");
iter.next();
iter.next();
iter.remove();
```

3.5

What is wrong with the `add()` method? The method is supposed to add an element to the end of a doubly linked list. Also, assume the constructor of a Node to be of the form

```
public Node(Integer d, Node p, Node n) {
    data = d;
    previous = p;
    next = n;
}


public class MyListImpl implements MyList {
    Node start;
    Node end;

    public void add(Integer i) {
        Node n = new Node(i, end, null);
        end = n;
        if (start == null)
            start = n;
        if (end != null)
            end.next = n;
    }
}
```

3.6

What is wrong with the `add()` method? The method is supposed to add an element to the end of a doubly linked list. Also, assume the constructor of a Node to be of the form

```
public Node(Integer d, Node p, Node n) {
    data = d;
    previous = p;
    next = n;
}

public class MyListImpl implements MyList {
    Node start;
    Node end;

    public void add(Integer i) {
        Node n = new Node(i, end, null);
        if (end != null)
            end.next = n;
        end = n;
    }
}
```

4 Stacks and Queues

4.1

What is the difference between a Java `Set` and a `List`?

4.2

Name three interfaces within the Java collections framework that directly extend the `Collection` interface.

4.3

Name two classes within the Java collections framework that implement the `List` interface, but no other interface besides the ones that `List` is extending.

4.4

Is `Stack` an interface or a class? What other interface or class does it directly extend or implement, and is that an interface or a class?

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 onto another stack, and taken out again. In which order do you now have these elements, and which order were they after each step?

4.6

Does the following code compile? If it does not, how can it be fixed? If it does compile, does it still contain an error? If so: how can it be fixed?

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

    public Node() {}
    public Node(Integer d, Node n) {
        data = d;
        next = n;
    }

    // Returns a String representing this, and following elements.
    public String nextString() {
        String ret = Integer.toString(data);
        return ret + ",␣" + next.nextString();
    }
}
```

```
public class Test {
  public static void main(String[] args) {}
}
```