CSC 1351: Exam On Everything

Name and Last Digit of Student ID: _____

1 Easy

1.1

What are the primitive types?
boolean, byte, char, short, int, long, float, double

2 Syntax

2.1

Write a HelloWorld program that prints "Hello World" to the screen.

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

2.2

Write a minimal Java program.

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

3 Inheritance

3.1

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?

```
class A {}
class B extends A {}
public class Main {
  public static void main(String[] args) {
    B b = new A();
  }
}
```

```
The code does not compile.

replace the code:     B b = new A();

with the code:     A a = new A();
```

3.2

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?

```
interface Planet { void orbit(); }
public class Mars implements Planet {
    public void orbit() { }
    public static void main(String[] args) {
        Planet a = new Planet();
        Mars b = new Mars();
    }}
```

The code does not compile.

replace the code: Planet a = new Planet();

with the code: Planet a = new Mars();

3.3

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?

```
class A {}
public class B extends A {
    public static void main(String[] args) {
        A a = new B();
        if(a instanceof B) b = a;
    } }
```

The code does not compile.

replace the code: if(a instanceof B) b = a;

with the code: B b = (B)a;

3.4

Fill in the missing code.

```
class A {}
public class B extends A {
    public static void main(String[] args) {
        A a = new B();
        B b = null;
        // ask if the value in a can be assigned to b
            b instanceof B
        if(----------------) {
            // set b using the value in a
            b = (B) a;
            --------------------
        }
    } }
```

3.5

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?

```
interface Moo {
  void moo(int a) {}
}
public class M implements Moo {
  public void moo(int a) {}
}
```

The code does not compile.

replace the code: void moo(int a) {}

with the code: void moo(int a);

3.6

Give then code below, write the code for the interface named Transformer.

```
public class Trans {
    public static void main(String[] args) {
        Transformer t = new Transformer() {
            public int transform(int n) { return n*2+1; }
        };
        System.out.println("t.transform(3)="+t.transform(3));
    }}

interface Transformer {
    int transform(int a);
}
```

3.7

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?

```
class Rocket {
  int energy = 3;
  void launch() { energy -= 2; check(); }
  void check()  {
    if(energy <= 0) System.out.println("out␣of␣power");
    else if(energy <= 1) System.out.println("low␣power");
  }}
public class Starship extends Rocket {
  void launch() { energy -= 5; check(); }
  public static void main(String[] args) {
    Rocket r1 = new Rocket();
    Rocket r2 = new Starship();
    r1.launch(); r2.launch();
  }}
```

low power
out of power

3.8

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?

```
class A {}
class B extends A {}
class C extends A,B {}
public class Main {
  public static void main(String[] args) {
    A a = new C();
    B b = new C();
  }
}
```

The code does not compile.

replace the code: class C extends A,B {}

with the code: class C extends B {}

4 Arrays

4.1

The code should print out 3, 9, 7, and 2 in sequence. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.Iterator;
class IntIterator implements Iterator<Integer> {
    int[] array; int index = 0;
    IntIterator(int[] array) {
        this.array = array;
        --------------------
    }
    public boolean hasNext() {
        return index < array.length;
        --------------------
    }
    public Integer next() {
        return array[index++];
    } }
public class MyIter {
    public static void main(String[] args) {
        IntIterator ii = new IntIterator(new int[]{3,9,7,2});
        while(ii.hasNext()) {
            int v = ii.next();
            System.out.println(v); } } }
```

4.2

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.util.Arrays;
public class Foo {
  public static void main(String[] args) {
    String[] letters = {"O","C","F","N","A","L","S","H","V","Y"};
    String result = letters[5] + letters[1];
    result = letters[4] + result;
    result += letters[0];
    result = letters[2] + result + letters[3];
    StringBuilder sb = new StringBuilder();
    sb.append(result);
```

```
    for(int i=7;i<letters.length;i++)
        sb.append(letters[i]);
    System.out.println(sb.toString()); } }
```

```
FALCONHVY
```

4.3

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?

```
public class ArraysInJava
{
    public static void main(String[] args)
    {
        int[] i = new int[0];
        System.out.println(i[0]);
    }
}
```

```
The code compiles, but when it runs it throws a ArrayIndexOutOfBoundsException

replace the code:          int[] i = new int[0];

with the code:          int[] i = new int[1];
```

4.4

Keep only array elements that are more than 3. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class MoreThan3 {
    public static int[] moreThan3(int[] a) {
        int n1 = 0, n2 = 0;
        --------------------
        for(int i=0;i<a.length;i++)
        --------------------
            if(a[i] > 3) n1++;
            --------------------
        int[] b = new int[n1];
        --------------------
        for(int i=0;i<a.length;i++)
        --------------------
            if(a[i] > 3) b[n2++] = a[i];
            --------------------
        return b;
        --------------------
    }
    public static void main(String[] args) {
        int[] r1 = new int[]{5,2,7,9,8,3}; int[] r2 = moreThan3(r1);
        System.out.println(Arrays.toString(r2));} }
```

// output:

```
[5, 7, 9, 8]
```

4.5

Write another version of the moreThan3 method that works with Lists.

4.6

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.util.Arrays;
public class Foo {
  public static void main(String[] args) {
    String[] letters = {"A","G","I","N","W","Y", "S"};
    String result = letters[5] + letters[1];
    result = letters[4] + result + letters[2];
    result = result + result.substring(0, 3) + letters[0];
    result = result + letters[2] + letters[3] + result.charAt(2) + result.charAt(0);
    System.out.println(result);
  }
}
```

WYGIWYGAINGW

4.7

Write a method that, given an array of integers and an integer s, prints all pairs of array elements whose sum is equal to s. Don't print other pairs, print pairs only once, and print the smaller number first. Assume the array does not contain duplicate entries.

4.7.1

What is the big-O speed of your solution, if N is the size of the array?

4.8

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;
    }
}
```

n is assigned to end before using it to assign end.next.

4.9

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?

```
public class Alist {
    int[] data = new int[]{3,4,5,9,7}; int size=4;
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append('{');
        for(int i=0;i<size;i++) {
            if(i > 0) sb.append(',');
            System.out.println("i="+i);
            sb.append(i);
        }
        sb.append('}');
        return sb.toString(); }
    public static void main(String[] args) {
        System.out.println(new Alist());
    } }
```

```
i=0
i=1
i=2
i=3
{0,1,2,3}
```

4.10

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.util.Arrays;
public class Foo {
  public static void main(String[] args) {
    String[] letters = {"G","A","I","N","Y","W","R"};
    String result = letters[4] + letters[0];
    result = letters[5] + result + letters[2];
```

```
    result = result + result.substring(0, 3) + letters[1];
    result = result + result.charAt(3) + letters[6] + letters[3] + result.charAt(2) +
       letters[5];
    System.out.println(result);
  }
}
```

WYGIWYGAIRNGW

4.11

Implement an ArrayList: Unlike an array, an ArrayList needs to be able to support appending elements. Usually, an ArrayList is implemented by maintaining an array that is larger than required, and another variable is used to keep track of how many elements have been stored in the array. Fill in the missing code for the add method, doing what is appropriate for a generic class.
Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class Alist {
  Object[] data = new Object[10];
  --------------------
  int size=0;
  public void add(Object item) {
    if(size >= data.length) {
      Object[] newData = new Object[data.length+10];
      --------------------
      for(int i=0;i<data.length;i++)
      --------------------
        newData[i] = data[i];
        --------------------
      data = newData;
      --------------------
    }
    data[size++] = item;
  }
}
```

4.12

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;
    }
}
```

`start` is not changed in case the list is empty.

4.13

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?

```
public class MyList {
    Object[] o = new Object[10];
    int size = 0;
    void add(Object d) { o[size++] = d; }
    Object get(int n) { assert n < size; return o[n]; }
    public static void main(String[] args) {
        MyList ml = new MyList();
        ml.add(3); ml.add(2); ml.add(1);
        System.out.println(ml.get(3));
    } }
```

```
The code compiles, but when it runs it throws a AssertionError

replace the code:          System.out.println(ml.get(3));

with the code:          System.out.println(ml.get(2));
```

4.14

Write a method that creates a reversed copy of an array. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.Arrays;
public class Reverse {
    static int[] reverse(int[] input) {
        int[] output = new int[input.length];
        --------------------
        int n = input.length-1;
        --------------------
        for(int i=0;i<input.length;i++)
        --------------------
            output[n--] = input[i];
            --------------------
        return output;
        --------------------
    }
    public static void main(String[] args) {
        int[] in = new int[]{3,8,2,4};
        int[] out = reverse(in);
        System.out.println(Arrays.toString(in));
        System.out.println(Arrays.toString(out));
    }
}
```

// output:

```
[3, 8, 2, 4]
[4, 2, 8, 3]
```

4.15

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.util.Arrays;
public class ReverseShift {
  // Shift values to the right, thus [0,1,2] should become [2,1,0]
```

```
  static void shift(int[] s) {
    int tmp = s[s.length-1];
    for(int i=0;i<s.length-1;i++)
      s[i+1] = s[i];
    s[0] = tmp;
  }
  public static void main(String[] args) {
    int[] n = new int[3];
    for(int i=0;i<n.length;i++) n[i] = i;
    shift(n);
    for(int i=0;i<n.length-1;i++)
      for(int j=i+1;j<n.length;j++)
        assert(n[i] != n[j]);
    System.out.println(Arrays.toString(n));
  }
}
```

The code compiles, but when it runs it throws a AssertionError

replace the code: `for(int i=0;i<s.length-1;i++)`

with the code: `for(int i=s.length-2;i>=0;i--)`

4.16

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?

```
public class Alist {
    Object[] data = new Integer[]{3,4,5,9,7}; int size=3;
    public void add(Object item) {
        if(size+1 >= data.length) {
            Object[] newData = new Object[data.length+10];
            for(int i=0;i<newData.length;i++)
                newData[i] = data[i];
            data = newData;
        }
        data[size++] = item; }
    public static void main(String[] args) {
        Alist a = new Alist();
        for(int i=0;i<20;i++) a.add(i); } }
```

The code compiles, but when it runs it throws a ArrayIndexOutOfBoundsException

replace the code: `for(int i=0;i<newData.length;i++)`

with the code: `for(int i=0;i<data.length;i++)`

4.17

Find the best match between the column on the left and the column on the right.

(1) boolean	3	(a) has no method bodies
(2) ArrayList	8	(b) converts one type to another
(3) interface	2	(c) has size() method
(4) String	7	(d) used to create a subclass
(5) Object	5	(e) default superclass
(6) built-in array	1	(f) a primitive type
(7) extends	6	(g) has fixed length
(8) casting	4	(h) has length() method

4.18

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.util.*;
public class Shift {
  // shift values to the left
  static void shift(List<Integer> s) {
    int tmp = s.get(0);
    for(int i=1;i<s.size();i++)
      s.set(i-1,s.get(i));
    s.set(s.size()-1,tmp);
  }
  public static void main(String[] args) {
    List<Integer> n = new ArrayList<>();
    for(int i=0;i<3;i++) n.add(i);
    shift(n);
    for(int i=0;i<n.size()-1;i++)
      for(int j=i+1;j<n.size();j++)
        assert(n.get(i) != n.get(j));
    System.out.println(n);
  }
}
```

```
[1, 2, 0]
```

4.19

Return true if a list contains, somewhere, three adjacent numbers that double each time .... 2, 4, 8, ... or 3, 6, 12.

```
tripleTrouble([7, 1, 2, 4, 5, 3]) -> true
tripleTrouble([2, 4, 8]) -> true
tripleTrouble([0, -1, -2]) -> false
tripleTrouble([0, 0, 0]) -> true
tripleTrouble([]) -> false
```

The function prototype is this: boolean tripleTrouble(List¡Integer¿ list) ... Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
  public static boolean tripleTrouble(List<Integer> array) {
    for(int i=0;i+2<array.size();i++) {
    ---------------------
      if(array.get(i)*2 == array.get(i+1) && array.get(i+1)*2 == array.get(i+2))
      ---------------------
        return true;
        ---------------------
    }
    ---------------------
    return false;
    ---------------------
  }
```

4.20

What is the advantage of using an array class? It allows for resizable arrays.

5 Anonymous Inner Class

5.1

Given the interface defined like this:

```
public interface MouseListener {
  void mousePressed(int x,int y);
  void mouseReleased(int x,int y);
}
```

Write a complete program that uses an anonymous inner class that implements this interface, then calls mousePressed() to print the x and y values of the mouse event to the screen.

```
public class Test {
  public static void main(String[] args) {
    MouseListener ml = new MouseListener() {
      public void mousePressed(int x,int y) {
        System.out.println("("+x+","+y+")");
      }
      public void mouseReleased(int x,int y) {
      }
    };
    ml.mousePressed(230,127);
  }
}
```

5.2

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

```
interface Operator { int eval(int a,int b); }
public class Anon {
  public static void main(String[] args) {
    Operator add =
        new Operator() {
        --------------------
            public int eval(int a,int b) {
            --------------------
                return a+b;
                --------------------
            }};
            --------------------
    Operator mul =
        new Operator() {
        --------------------
            public int eval(int a,int b) {
            --------------------
                return a*b;
                --------------------
            }};
            --------------------
    System.out.println("add="+add.eval(2,3)+"␣mul="+mul.eval(2,3));
  }
}
```

// output:

```
add=5 mul=6
```

5.3

Given the interface defined like this:

```
public interface KeyListener {
  void keyPressed(char key);
  void keyReleased(char key);
}
```

Write a complete program that uses an anonymous inner class that implements this interface, then calls keyPressed() to print the key character 'A' to the screen.

```
public class Test {
  public static void main(String[] args) {
    KeyListener kl = new KeyListener() {
      public void keyPressed(char c) {
        System.out.println(c); }
      public void keyReleased(char c) {
        System.out.println(c); }
    };
    kl.keyPressed('A'); } }
```

5.4

The following program should print the string "Run!" to the screen.
Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
interface RunMe { void run(); }
public class Runner {
    public static void main(String[] args) {
        RunMe rm =
            new RunMe() {
            --------------------
                public void run() {
                --------------------
                    System.out.println("Run!");
                    --------------------
                }};
                --------------------
        rm.run(); }}
```

// output:

```
Run!
```

5.5

Figure out the anonymous inner class Fill in the missing code.

```
import java.util.*;
public class Sorter {
  public static void main(String[] args) {
    Random rand = new Random();
    List<Integer> li = new ArrayList<>();
    for(int i = 0;i< 10;i++)
      li.add(rand.nextInt(20));
    // Comparator<Integer> has a single method
    // public int compare(Integer a,Integer b)
    // which must be supplied.
                            new Comparator<Integer>() {
    Comparator<Integer> c = --------------------------
      public int compare(Integer a,Integer b) {
      --------------------
        return b - a;
```

```
        --------------------
      }
      --------------------
    };
    --------------------
    Collections.sort(li,c);
    System.out.println(li);
    System.out.println(c.compare(3,9));
  }
}
```

5.6

Given the interface defined like this:

```
public interface Logger {
  void log(String msg);
}
```

Write a complete program that uses an anonymous inner class that implements this interface to print the message msg to the screen.

```
interface Logger {
  void log(String msg);
}
public class Test {
  public static void main(String[] args) {
    Logger log = new Logger() {
      public void log(String msg) {
        System.out.println(msg);
      }
    };
    log.log("Hello␣World");
  }
}
```

5.7

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
```

5.8

Fill in the missing code.

```
interface GetInt {
   public int value();
}
public class Anon2 {
  public static void main(String[] args) {
                new GetInt() {
    GetInt gi = --------------
      public int value() {
      --------------------
        return 33;
        --------------------
      }
      --------------------
    };
    System.out.println("value="+gi.value());
  }
}
```

// output:

```
value=33
```

6 Recursion

6.1

Rewrite the following code to use recursion (no loops!).

```
public class Hello {
  public static void hello(int n) {
    for(int i=0;i<n;i++)
      System.out.println("Hello␣"+i);
  }
  public static void main(String[] args) {
    hello(3);
  }
}
```

```
public class Hello {
  public static void hello(int n) {
    hello(0,n);
  }
  public static void hello(int i,int n) {
    if(i<n) {
      System.out.println("Hello␣"+i);
      hello(i+1,n);
    }
  }
  public static void main(String[] args) {
    hello(3);
  }
}
```

6.2

From CodingBat.
Given a string, compute recursively a new string where all the adjacent chars are now separated by a ”*”.

```
allStar("hello") -> "h*e*l*l*o"
allStar("abc") -> "a*b*c"
allStar("ab") -> "a*b"
```

```
  public static String allStar(String str) {
    return allStar(0,str);
  }
  public static String allStar(int n,String str) {
    if(n == str.length()) {
      return "";
    } else if(n+1 == str.length()) {
      return ""+str.charAt(n);
    } else {
      return ""+str.charAt(n)+"*"+allStar(n+1,str);
    }
  }
```

6.3

What is wrong with the following program, what happens when you run it? If this method would have been implemented recursively, and would have a similar error, what would happen then if run?

```
public class JavaIsToJavascriptWhatCarIsToCarpet {
  public static int factorial(int n) {
    int result = 1;
    while (n > 1) {
      result *= n;
    }
    return result;
  }
  public static void main(String args[]) {
    assert(factorial(4) == 24);
  }
}
```

There is an exit-condition missing in the while loop. When run, the program hangs. This is similar to infinite recursions, so the same in a recursive program would result in a stack overflow.

6.4

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?

```
public class Series {
  public static int func(int j) {
  System.out.println(j);
  if (j==1) return 1;
    return 2*func(j-1) + 5*func(j-2); }
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    if (N<1) {
      System.out.println("invalid␣argument");
      return;
    }
    System.out.println(func(N)); } }
```

```
The code compiles, but when it runs it throws a StackOverflowError

replace the code:   if (j==1) return 1;

with the code:   if (j<1) return 1;
```

6.5

Martian mathematicians have uncovered a mysterious recursive formula. Implement it below. No loops allowed.
MarsCode a:

1. if a $< 10 \rightarrow 10$
2. if a is odd $\rightarrow$ three times the value MarsCode ((a+1)/2) plus the value of MarsCode ((a-1)/2)
3. if a is even $\rightarrow$ two times the value of MarsCode (a/2)

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

```
public class MarsCode {
    public static int eval(int a) {
        if(a < 10) return 10;
        --------------------
        if(a % 2 == 1) return 3*eval((a+1)/2) + eval((a-1)/2);
        --------------------
        return 2*eval(a/2);
        --------------------
    }
    public static void main(String[] args) {
        for(int i=20;i<25;i++) {
            System.out.println("eval("+i+")="+eval(i));
        }
    }
}
```

// output:

```
eval(20)=40
eval(21)=140
eval(22)=80
eval(23)=100
eval(24)=40
```

6.6

From CodingBat:
Given a string, compute recursively a new string where all the lowercase 'x' chars have been moved to the end of the string.

```
endX("xxre") -> "rexx"
endX("xxhixx") -> "hixxxx"
endX("xhixhix") -> "hihixxx"

  public static String endX(String x) {
    return endX(0,x);
  }
  public static String endX(int n,String x) {
    if(n == x.length()) return "";
    if(x.charAt(n) == 'x')
      return ""+endX(n+1,x)+"x";
    return ""+x.charAt(n)+endX(n+1,x);
  }
```

6.7

You want to implement flipping a coin on a computer. However, you want to be fancy and include the possibility of the coin standing on edge, with a probability of 2%. In that case you flip it again, until the coin lands on either side. "Heads" give 1 point, "Tails" give 0 points, and an edge gives an additional 2 points (potentially multiple times). Most of the time the return value will be 0 or 1. However, if the coin landed on edge once, the return value will be either 2 or 3, depending on the final toss. If it landed on edge twice, it could be either 4 or 5, in a similar way. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
  final static Random rand = new Random();
  public static int flip() {
    int randomPercentage = rand.nextInt(100);
    if (randomPercentage <  2) return 2 + flip();
    --------------------
    if (randomPercentage < 51) return 0;
    --------------------
    return 1;
    --------------------
  }
```

6.8

This task involves a game with plush owls. You start the game with a given number of owls. You can then give back some of the owls, according to the rules below. If multiple rules apply, it is your choice which to use. If none applies, you loose. The goal is to end up with exactly 42 owls. n is the number of owls you currently have.

- If n is even, you may give back exactly n/2 owls.
- If n is divisible by 3 or 4, you may multiply the last two digits of n and give back this many owls. (the last digit of n is n%10, the next-to-last digit is ((n%100)/10).
- If n is divisible by 5, then you may give back exactly 42 owls.

For example, suppose you start with 250 owls, you could do the following moves:

- Since 250 is divisible by 5, you may return 42 owls, leaving you with 208.
- Since 208 is even, you may return half of them, leaving you with 104.
- Since 104 is even again, you may do the same again, leaving you with 52.
- Since 52 is didisible by 4, you may multiply the last two digits (2*5=10), and return 10 owls, leaving you with 42 owls, and resulting in a win.

Write a recursive function to meet this specification:

```
// The return value should indicate whether it is possible to win this game if
// you start with n owls. For example:
// owls(250) is true
// owls(42) is true
// owls(84) is true
// owls(53) is false
// owls(41) is false
public class Owls {
    public static boolean owls(int n) {
        // smaller than 42
        if (n<42) return false;
        // equal 42
        if (n==42) return true;
        // divisible by 2
        if (n%2==0 && owls(n/2)) return true;
        // divisible by 3 or 4
        if ((n%3==0||n%4==0) && owls(n-((n%10)*((n%100)/10)))) return true;
        // divisible by 5
        if (n%5==0 && owls(n-42)) return true;
        return false;
    }
```

6.9

The following program determines whether exactly 3 of the supplied values can be combined to produce the target value. Fill in the missing code. Use recursion. No loops.
Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class GroupSum3 {
    public static boolean sum3(int[] values,int target) {
        return sum3(0,values,target,3);
    }
    public static boolean sum3(int start,int[] values,int target,int count) {
        if(start >= values.length) return target == 0 && count == 0;
        --------------------
        boolean ret = sum3(start+1,values,target-values[start],count-1);
        --------------------
        if(!ret) ret = sum3(start+1,values,target,count);
        --------------------
        return ret;
        --------------------
    }
}
```

6.10

What is wrong with the following program, what happens when you run it? If this method would have been implemented using a loop, and would have a similar error, what would happen then if run?

```
public class JavaIsToJavascriptWhatCarIsToCarpet {
  public static int factorial(int n) {
    return(n * factorial(n-1));
  }
  public static void main(String args[]) {
    factorial(4);
  }
}
```

There is an exit-condition missing in the recursive function. When run, you get a stack overflow. This is similar to infinite loops, so the same in an iterative program would result in a hang during execution.

6.11

What is wrong with the following program, what happens when you run it? If this method would have been implemented recursively, and would have a similar error, what would happen then if run?

```
public class JavaIsToJavascriptWhatCarIsToCarpet {
  public static int factorial(int n) {
    int result = 1;
    while (n > 1) {
      result *= n;
    }
    return result;
  }
  public static void main(String args[]) {
    assert(factorial(4) == 24);
  }
}
```

There is an exit-condition missing in the while loop. When run, the program hangs. This is similar to infinite recursions, so the same in a recursive program would result in a stack overflow.

6.12

Implement the factorial() method in the JavaIsToJavascriptWhatCarIsToCarpet class above recursively, correctly.

```
public static int factorial(int n) {
  if (n < 1) return 1;
  return(n * factorial(n-1));
}
```

6.13

Write a recursive program to compute a double factorial, n!! = n*(n-2)!!, 0!! = 1, 1!! = 1.

```
public class DFac {
    static int dfac(int n) {
        if(n < 2) return 1;
        return n*dfac(n-2);
    }
    public static void main(String[] args) {
        assert dfac(7) == 7*5*3*1; } }
```

6.14

What is recursion?

1. Recursion is a generic class.
2. Recursion is a process of setting a value based on it's previous value.
3. Recursion is a process of defining a method that calls itself.
4. Recursion is a process of repeatedlty calling other methods.

Recursion is a process of defining a method that calls itself.

6.15

You want to implement combat within a role playing game on a computer. Specifically, the game rules for damage inflicted by a hit are:

- In order to figure out damage from one hit, you throw a N-sided die.
- The result of one throw will be between 1 and N (including both, e.g., a 6-sided die has six sides, labled 1 to 6).
- If the result is 1 to N-1, that is the resulting damage from the hit.
- If the result in N, however, you hit critically, and you throw again, adding the results.
- If you throw again, the same rules apply, potentially resulting in doubly or more critical hits.

For example, if you use a 4-sided die and throw a 3, the damage is 3. If you throw a 4 instead, you throw again. If that results in a 3, the total damage is 7. If you happen to throw two 4s after each other and then a 2, the total damage is 10. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
final static Random rand = new Random();
public static int damage(int n) {
  int result = rand.nextInt(n) + 1;
  --------------------
  if (result < n) return result;
  --------------------
  return result + damage(n);
  --------------------
}
```

6.16

What is the output of this program?

```
class recursion {
    int func (int n) {
        int result;
        result = func (n - 1);
        return result;
    }
}
public class Output {
    public static void main(String args[]) {
        recursion obj = new recursion() ;
        System.out.print(obj.func(12));
    }
}
```

1. 0
2. 1
3. 12
4. Compilation Error
5. Runtime Error

Runtime Error: Exception in thread main java.lang.StackOverflowError

6.17

Given an List of Integers, is it possible to choose a group of Integers, such that the sum of all Integers in that group matches a given target with this additional constraint: The number of integers chosen must be odd.
Do not use loops.
Example inputs and returns for `groupOdd()`:

`[],0`	false
`[1],1`	true
`[1,1,3], 2`	false
`[1,1,3], 4`	false
`[1,1,3], 5`	true
`[1,2,3], 4`	false

Fill in the missing code.

```
  public static boolean groupOdd(List<Integer> list, int target) {
    return groupOdd(0,list,target,0);
    --------------------
  }
                                  int start,List<Integer> list, int target,int num
  public static boolean groupOdd(--------------------------------------------------) {
    if (start >= list.size()) return target == 0 && (num % 2)==1;
    --------------------
    int val = list.get(start);
    --------------------
    boolean works = groupOdd(start+1,list,target-val,num + 1);
    --------------------
    if(!works)
    --------------------
        works = groupOdd(start+1,list,target,num);
        --------------------
    return works;
  }
```

6.18

Given an List of Integers, is it possible to choose a group of Integers, such that the sum of all Integers in that group matches a given target with this additional constraint: If a value in the List is chosen to be in the group, the value immediately following it in the List must not be chosen. Do not use loops.
Example inputs and returns for `groups()`:

`[],0`	false
`[1],0`	false
`[1],1`	true
`[1,2], 1`	true
`[1,2], 3`	false
`[1,0], 1`	true
`[1,0], 0`	true
`[1,2,3], 4`	true

You may (but don't have to) use the `subList` method, defined in `java.util.List`, which is declared as follows, and returns a part (slice) of the input array (fromIndex inclusive, toIndex exclusive):

```
List<E> subList(int fromIndex, int toIndex)
```

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

```
public static boolean groups(List<Integer> list, int target) {
  if (list.size() == 0) return false;
  --------------------
  if (list.get(0) == target) return true;
  --------------------
  if (list.size() >= 2 &&
  --------------------
      groups(list.subList(2, list.size()), target-list.get(0)) == true) return true;
      --------------------
  return groups(list.subList(1, list.size()), target);
  --------------------
}
```

6.19

This task involves a game with plush bunnies. You start the game with a given number of bunnies. You can then give back some of the bunnies, according to the rules below. If multiple rules apply, it is your choice which to use. If none applies, you loose. The goal is to end up with exactly 42 bunnies. n is the number of bunnies you currently have.

- If n is even, you may give back exactly n/2 bunnies.
- If n is divisible by 3 or 4, you may multiply the last two digits of n and give back this many bunnies. (the last digit of n is n%10, the next-to-last digit is ((n%100)/10).
- If n is divisible by 5, then you may give back exactly 42 bunnies.

For example, suppose you start with 250 bunnies, you could do the following moves:

- Since 250 is divisible by 5, you may return 42 bunnies, leaving you with 208.
- Since 208 is even, you may return half of them, leaving you with 104.
- Since 104 is even again, you may do the same again, leaving you with 52.
- Since 52 is didisible by 4, you may multiply the last two digits (2*5=10), and return 10 bunnies, leaving you with 42 bunnies, and resulting in a win.

Write a recursive function to meet this specification:

```
// The return value should indicate whether it is possible to win this game if
// you start with n bunnies. For example:
// bunnies(250) is true
// bunnies(42) is true
```

```
// bunnies(84) is true
// bunnies(53) is false
// bunnies(41) is false
public class Bunnies {
    public static boolean bunnies(int n) {
        // smaller than 42
        if (n<42) return false;
        // equal 42
        if (n==42) return true;
        // divisible by 2
        if (n%2==0 && bunnies(n/2)) return true;
        // divisible by 3 or 4
        if ((n%3==0||n%4==0) && bunnies(n-((n%10)*((n%100)/10)))) return true;
        // divisible by 5
        if (n%5==0 && bunnies(n-42)) return true;
        return false;
    }
```

6.20

Which of these will happen if recursive method does not have a base case?

1. An infinite loop occurs, hanging forever.
2. The program stops after some time with an error.
3. After 1000000 calls it will be automatically stopped.
4. None of the mentioned

The program stops after some time with an error. (stack overflow)

6.21

```
public class Exp {
  // Compute the exponential function
  public static double exp(double x) {
    if(x < .001) {
      return 1+x+x*x/2+x*x*x/6;
    } else {
      double ex = exp(0.5*x);
      return ex*ex;
    }
  }
  public static void main(String[] args) {
    for(double x=0;x<10;x+=2.0) {
      System.out.printf("exp(%f)=%f␣(err=%f)%n",x,exp(x),Math.abs(Math.exp(x)-exp(x)))
          ;
    }
  }
}
```

// output:

```
exp(0.000000)=1.000000 (err=0.000000)
exp(2.000000)=7.389056 (err=0.000000)
exp(4.000000)=54.598150 (err=0.000000)
exp(6.000000)=403.428793 (err=0.000000)
exp(8.000000)=2980.957986 (err=0.000001)
```

7 Sort and Search

7.1

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

```
public class LunarLander implements Comparable {
  boolean manufacturedOnMars; // sort by this first, true is higher priority
  String name; // sort by this second
  int energyStorage; // sort by this third
  @Override
  public int compareTo(Object o) {
    LunarLander that = (LunarLander)o;
    --------------------
    int diff = 0;
    --------------------
    if(this.manufacturedOnMars) diff -= 1;
    --------------------
    if(that.manufacturedOnMars) diff += 1;
    --------------------
    if(diff == 0) diff = this.name.compareTo(that.name);
    --------------------
    if(diff == 0) diff = this.energyStorage - that.energyStorage;
    --------------------
    return diff;
    --------------------
  }
}
```

7.2

A special sort routine can be written if we are sorting integers in the range 0 to N-1. It looks like this:

```
  public static void sort(int N,int[] values) {
    int[] boxes = new int[N]; int n = 0;
    for(int i=0;i<values.length;i++)
      boxes[values[i]]++; // put each item into its own box
    for(int i=0;i<boxes.length;i++) {
      while(boxes[i] > 0) { // while each box is not empty
        values[n++] = i; // move an item from the box in the array
        boxes[i]--; } } }
```

Does it work? What is its big O behavior in terms of the length of the array L? Is it O(L log L)? O(L^2)? Something else? Justify your answer.
Yes, it works. This is a bucket sort and is O(L). This is plain because the first loop manifestly iterates over the array once. The double loop iterates over exactly L items, as it refills the array without causing an index exception.

7.3

What is the time complexity of each of these sort methods in terms of the array size, N?

1. Bubble Sort
2. Selection Sort
3. Merge Sort
4. Quick Sort

1. Bubble Sort $O(N^2)$

2. Selection Sort $O(N^2)$
3. Merge Sort $O(N \log N)$
4. Quick Sort $O(N \log N)$

7.4

What kind of sort is this? Rewrite it to use ArrayList.

```
static void sort(int end,int[] values) {
  if(end < 2)
    return;
  for(int i=1;i<end;i++) {
    if(values[i-1] > values[i]) {
      int tmp = values[i];
      values[i] = values[i-1];
      values[i-1] = tmp;
    }
  }
  sort(end-1,values);
}
```

This is a bubble sort because it's constantly comparing adjacent elements.

```
static void sort(int end,ArrayList<Integer> values) {
  if(end < 2)
    return;
  for(int i=1;i<end;i++) {
    if(values.get(i-1) > values.get(i)) {
      int tmp = values.get(i);
      values.set(i,values.get(i-1));
      values.set(i-1,tmp);
    }
  }
  sort(end-1,values);
}
```

7.5

What kind of sort is this? Rewrite it to use ArrayList.

```
static void sort(final int start,final int end,int[] values) {
  if(end-start < 1)
    return;
  int[] scratch = new int[end-start+1];
  int r = rand.nextInt(scratch.length)+start;
  int p = values[r];
  int lo = 0, hi = scratch.length;
  int numps = 0;
  for(int i=start;i<=end;i++) {
    if(values[i] < p)
      scratch[lo++] = values[i];
    else if(values[i] == p)
      numps++;
    else
      scratch[--hi] = values[i];
  }
  int n = start;
  for(int i=0;i<lo;i++)
    values[n++] = scratch[i];
  for(int i=0;i<numps;i++)
```

```
      values[n++] = p;
    for(int i=hi;i<scratch.length;i++)
      values[n++] = scratch[i];
    sort(start,start+lo-1,values);
    sort(start+hi,end,values);
  }
```

The random number generation gives it away. This is a quicksort.

```
  static void sort(final int start,final int end,ArrayList<Integer> values) {
    if(end-start < 1)
      return;
    int[] scratch = new int[end-start+1];
    int r = rand.nextInt(scratch.length)+start;
    int p = values.get(r);
    int lo = 0, hi = scratch.length;
    int numps = 0;
    for(int i=start;i<=end;i++) {
      if(values.get(i) < p)
        scratch[lo++] = values.get(i);
      else if(values.get(i) == p)
        numps++;
      else
        scratch[--hi] = values.get(i);
    }
    int n = start;
    for(int i=0;i<lo;i++)
      values.set(n++,scratch[i]);
    for(int i=0;i<numps;i++)
      values.set(n++,p);
    for(int i=hi;i<scratch.length;i++)
      values.set(n++,scratch[i]);
    sort(start,start+lo-1,values);
    sort(start+hi,end,values);
  }
```

7.6

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

```
public class Bug implements Comparable {
  String type;  // sort by this if num_legs and num_eyes are equal
  int num_legs; // sort by this first
  int num_eyes; // sort by this if num_legs is equal
  @Override
  public int compareTo(Object o) {
    Bug that = (Bug)o;
    --------------------
    int diff = this.num_legs - that.num_legs;
    --------------------
    if(diff != 0) return diff;
    --------------------
    diff = this.num_eyes - that.num_eyes;
    --------------------
    if(diff != 0) return diff;
    --------------------
    return this.type.compareTo(that.type);
    --------------------
  }
}
```

7.7

Make the above code generic.

```
public class Bug implements Comparable<Bug> {
  String type;  // sort by this if num_legs and num_eyes are equal
  int num_legs; // sort by this first
  int num_eyes; // sort by this if num_legs is equal
  @Override
  public int compareTo(Bug that) {
    int diff = this.num_legs - that.num_legs;
    if(diff != 0) return diff;
    diff = this.num_eyes - that.num_eyes;
    if(diff != 0) return diff;
    return this.type.compareTo(that.type);
  }
}
```

7.8

What kind of sort is this? Something was done incorrectly, however. What is wrong? Fix it.

```
  static void sort(int[] values) {
    sort(values,0,values.length); }
  static void sort(int[] values,int start,int end) {
    if(end - start < 2) return;
    int mval = values[(start+end)/2];
    int i1 = start, i2 = end;
    for(int i=start;i<end;i++) {
        if(values[i] < mval) {
            int tmp = values[i];
            values[i] = values[i1];
            values[i1] = tmp; i1++; } }
    for(int i=end-1;i>=start;i--) {
        if(values[i] > mval) {
            i2--; int tmp = values[i];
            values[i] = values[i2];
            values[i2] = tmp; } }
    assert i1 < i2 : "i1="+i1+" i2="+i2+" start="+start+" end="+end  ;
    sort(values,start,i1); sort(values,i2,end);
  }
```

This is a quick sort because it selects a pivot and puts values before and after the pivot at different ends of the array. The pivot, however, should be chosen randomly.

```
Random RAND = new Random();
int mval = values[RAND.nextInt(values.length)];
```

7.9

What kind of sort is this? Rewrite it to use ArrayList.

```
  static void sort(int start,int[] values) {
    if(values.length-start < 2)
      return;
    int m = start;
    for(int i=start+1;i<values.length;i++)
      if(values[i] < values[m])
        m = i;
    int tmp = values[m];
    values[m] = values[start];
    values[start] = tmp;
```

```
    sort(start+1,values);
  }
```

This is a selection sort because it keeps finding the minimum and moving it to the start.

```
  static void sort(int start,ArrayList<Integer> values) {
    if(values.size()-start < 2)
      return;
    int m = start;
    for(int i=start+1;i<values.size();i++)
      if(values.get(i) < values.get(m))
        m = i;
    int tmp = values.get(m);
    values.set(m,values.get(start));
    values.set(start,tmp);
    sort(start+1,values);
  }
```

7.10

Find the best match between the column on the left and the column on the right.

(1) bubble sort	1	(a) compares adjacent elements
(2) quick sort	3	(b) can always be rewritten as recursive
(3) iterative	2	(c) uses a random number
(4) binary search	7	(d) keeps finding the next smallest element
(5) merge sort	5	(e) is $O(N \log N)$
(6) recursion	4	(f) is $O(\log N)$
(7) selection sort	6	(g) can cause StackOverflowError

7.11

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

```
public class Bug implements Comparable {
  int num_legs; // sort by this first
  int num_eyes; // sort by this if num_legs is equal
  @Override
  public int compareTo(Object o) {
    Bug that = (Bug)o;
    --------------------
    int diff = this.num_legs - that.num_legs;
    --------------------
    if(diff != 0) return diff;
    --------------------
    return this.num_eyes - that.num_eyes;
    --------------------
  }
  public static void main(String[] args) {}
}
```

7.12

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

```
public class Cat implements Comparable {
  int numKittens; // sort by this if the names are equal
  String name; // sort by this first
  @Override
  public int compareTo(Object o) {
    Cat that = (Cat)o;
```

```
    --------------------
    int diff = this.name.compareTo(that.name);
    --------------------
    if(diff != 0) return diff;
    --------------------
    return this.numKittens - that.numKittens;
    --------------------
  }
  public static void main(String[] args) {}
}
```

7.13

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

```
public class MarsLander implements Comparable {
  boolean canTakeOff; // sort by this first, true is higher priority
  String homeCountry; // sort by this second
  int fuelCap; // sort by this third
  @Override
  public int compareTo(Object o) {
    MarsLander that = (MarsLander)o;
    --------------------
    int diff = 0;
    --------------------
    if(this.canTakeOff) diff -= 1;
    --------------------
    if(that.canTakeOff) diff += 1;
    --------------------
    if(diff == 0) diff = this.homeCountry.compareTo(that.homeCountry);
    --------------------
    if(diff == 0) diff = this.fuelCap - that.fuelCap;
    --------------------
    return diff;
    --------------------
  }
}
```

7.14

What kind of sort is this? Something was done incorrectly, however. What is wrong? Fix it.

```
  public static Random RAND = new Random();
  static void sort(int[] values) {
    sort(values,0,values.length); }
  static void sort(int[] values,int start,int end) {
    if(end - start < 2) return;
    int mid = RAND.nextInt(end-start)+start;
    sort(values,start,mid);
    sort(values,mid,end);
    int[] newvalues = new int[end-start];
    int i1=start, i2=mid, i3=0;
    while(i1 < mid && i2 < end) {
        if(values[i1] <= values[i2])
            newvalues[i3++] = values[i1++];
        else
            newvalues[i3++] = values[i2++];
    }
    while(i1 < mid) newvalues[i3++] = values[i1++];
    while(i2 < end) newvalues[i3++] = values[i2++];
```

```
    for(int i=0;i<newvalues.length;i++)
        values[i+start] = newvalues[i];
  }
```

This is a merge sort because it does the recursive calls early on. It should, however, choose the midpoint rather than a random value.

```
int mid = (start+end)/2;
```

8 Collections

8.1

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`.

8.2

Name four interfaces within the Java collections framework.
List, Queue, Set, Collection, (SortedSet, NavigableSet, Deque, Iterable)

8.3

The program below is intended to prove that radar is a palindrome. If the assertion fails, it fails. 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.util.*;
public class Palindrome {
  public static void main(String[] args) {
    String pal = "radar";
    Queue<Character> queue = new LinkedList<>();
    Stack<Character> stack = new Stack<>();
    for (int i=0; i<pal.length(); i++) {
      char c = pal.charAt(i);
      queue.add(c);
      stack.push(c);
    }
    while (queue.size()>0) {
      Character st = stack.remove();
      Character qu = queue.remove();
      assert st.equals(qu);
    }
  }
}
```

```
The code does not compile.

replace the code:       Character st = stack.remove();

with the code:       Character st = stack.pop();
```

8.4

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) {}
}
```

The test for the end of the list is missing in `nextString()`.

```
        if (next != null)
            return ret + ",␣" + next.nextString();
        else
            return ret;
```

8.5

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

8.6

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();
```

*SPRING
S*PRING
*PRING
P*RING
*RING

R*ING
RA*ING
RAI*NG
RAIN*G
RAING*
RAIN*

8.7

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

8.8

Name two classes within the Java collections framework that implement the `List` interface, but no other interface besides the ones that `List` is extending.
ArrayList, Vector (but **not** LinkedList)

8.9

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

8.10

The program below is intended to prove that radar is a palindrome. If the assertion fails, it fails. 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.util.*;
public class Palindrome {
  public static void main(String[] args) {
    String pal = "radar";
    Queue<Character> queue = new LinkedList<>();
    Stack<Character> stack = new Stack<>();
    for (int i=0; i<pal.length(); i++) {
      char c = pal.charAt(i);
      queue.add(c);
      stack.push(c);
    }
    while (queue.size()>0) {
      Character st = stack.pop();
      Character qu = queue.remove();
      assert !st.equals(qu);
    }
  }
}
```

The code compiles, but when it runs it throws a AssertionError

replace the code: `assert !st.equals(qu);`

with the code: `assert st.equals(qu);`

8.11

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?
after queue: 2, 4, 6, 8 (original)
after stack: 8, 6, 4, 2 (reverse)
after stack: 2, 4, 6, 8 (original)

8.12

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?
`Stack` is a class, extending the class `Vector`.

8.13

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

8.14

Which class is `Stack` inheriting from, and which interface is that class implementing?
`Vector`, which is implementing `List`.

8.15

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)

8.16

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

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

public class MyList {
    Node start;
    Node end;

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

8.17

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 LinkedList<>();
    static Stack<String> operators = new LinkedList<>();
    static void evaluateTop() {
        Double n2 = numbers.pop();
        Double n1 = numbers.pop();
        String op = operators.pop();
        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);
    }
```

A `Stack` is not a subclass or implementation of `LinkedList`, but it's own class. Use `Stack` instead of `LinkedList`.

8.18

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

8.19

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.util.Stack;
public class Calc {
    Stack<Character> ops = new Stack<>();
    Stack<Integer> vals = new Stack<>();
    public Calc() {
        ops.push('+'); ops.push('-'); ops.push('+');
        vals.push(1); vals.push(2); vals.push(3); vals.push(4); }
    Integer eval() {
        while(!ops.isEmpty()) {
            Character c = ops.pop();
            Integer v1 = vals.pop();
            Integer v2 = vals.pop();
            if(c == '+') vals.push(v1+v2);
            else if(c == '-') vals.push(v1-v2);
        }
        return vals.pop(); }
    public static void main(String[] args) {
        Calc c = new Calc();
        System.out.println("c="+c.eval());
    } }
```

```
c=6
```

8.20

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();
```

```
*SPRING
S*PRING
SP*RING
S*RING
SR*ING
S*ING
SI*NG
S*NG
SU*NG
SUN*G
SUNG*
SUN*
```

8.21

Name three interfaces within the Java collections framework that directly extend the `Collection` interface.
List, Queue, Set

8.22

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.

8.23

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;
    }
}
```

8.24

A code that checks to see if W is a palindrome puts every character in W into both a stack and a queue (using push() and add()). It then repeatedly takes characters out of the stack and queue (using pop() and remove()) and compares them until

the stack and queue are empty. If all the pairs of characters are equal, then W is a palindrome. Why does this work?
Because a stack is a LIFO (a last-in-first-out), putting items into it and taking them out reverses the order. Because a queue is a FIFO (a first-in-first-out), putting items into it and taking them out preserves the order. A palindrome is the same backwards and forwards, and this procedure ensures that.

9 Time Complexity (Big O)

9.1

The following table gives approximate running times for a program with N inputs for various values of N.

N	time				
1000				8	seconds
2000	1:04	minutes	≈	64	seconds
5000	15	minutes	≈	1000	seconds
10000	2.2	hours	≈	8000	seconds

Which of the following best describes the likely running time of this program for N = 20,000?

1. A few hours (6,400 seconds)
2. About a day (64,000 seconds)
3. A few days (256,000 seconds)
4. About a week (640,000 seconds)

The scaling seems to be approximately N^3, so the estimated runtime would be $20^3 * 8$ seconds = 18 hours: about a day.

9.2

What is the Big-O notation for the following function?

$$T(N) = 1000N + 0.0003N^2 \tag{1}$$

The leading term as N gets large is N^2, so...
$O(N^2)$

9.3

What is the Big O speed for a linear search? For a merge sort? For an insertion sort?
For linear search it's $O(N)$.
For merge sort it's $O(N \log N)$
For insertion sort it's $O(N^2)$

9.4

What is the Big-O notation for the following function?

$$T(N) = N(N^2 - N \log N) \tag{2}$$

The leading term as N gets large is NN^2, so...
$O(N^3)$

9.5

Find the best match between the column on the left and the column on the right.

(1)	O(n^2)	6	(a) linear search
(2)	Requires a sorted list	4	(b) insertion sort
(3)	O(n log n)	1	(c) bubble sort
(4)	Repeatedly find the smallest value	5	(d) quick sort
(5)	Uses a random number	3	(e) merge sort
(6)	O(n)	2	(f) binary search

9.6

What is the Big-O notation for the following function?
$T(N) = 100000 + 10N + N^{-2}$
The leading term as N gets large is N, so...
$O(N)$

9.7

What is the Big-O notation for the following function?
$T(N) = 100000 + 10N - 3 + N^2$
The leading term as N gets large is N^2, so...
$O(N^2)$

9.8

What is the Big-O notation for the following function?

$$T(N) = 3N + \frac{37N + 93\sqrt{N^7}}{N^2} \tag{3}$$

The leading term as N gets large is $N^{\left(\frac{7}{2}-2\right)}$, so...
$O(N^{\frac{3}{2}})$

9.9

What is the Big O speed for a binary search? For a bubble sort? For a quick sort?
For binary search it's $O(\log N)$.
For bubble sort it's $O(N^2)$
For quick sort it's $O(N \log N)$

9.10

What is the Big-O notation for the following function?
$T(N) = 100000N^3 + 10N + N^{-2}$
The leading term as N gets large is N^3, so...
$O(N^3)$

9.11

The following table gives approximate running times for a program with N inputs for various values of N.

N		time
1000	5	seconds
2000	20	seconds
5000	2	minutes
10000	8	minutes

Which of the following best describes the likely running time of this program for N = 100, 000?

1. A few minutes
2. A few hours
3. Half a day
4. A few days

The scaling seems to be approximately N^2, so the estimated runtime would be $10^2 * 8$ minutes = 800 minutes: about half a day.

10 Exceptions

10.1

What is the Mars location 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?

```
public class Mars {
  String location = "Valles␣Marineris"; double fuel = 10;
  void travel(double fuelUsed,String location) throws Exception {
    if(fuelUsed > fuel) {
        fuel = 0; throw new Exception();
    } else {
        fuel -= fuelUsed; location = "Gale␣Crater"; } }
  public String toString() { return "{{"+location+":"+fuel+"}}"; }
  public static void main(String[] args) throws Exception {
    Mars m1 = new Mars(); Mars m2 = new Mars();
    try {
      m1.travel(9.5,"Gale␣Crater");
      m2.travel(10.5,"Meducae␣Fossae");
    } catch(Exception e) {
      System.out.println(m1);
      System.out.println(m2);
    } finally {
      System.out.println("travel␣complete");
    }
    System.out.println("Success"); } }
```

```
{{Valles Marineris:0.5}}
{{Valles Marineris:0.0}}
travel complete
Success
```

I'l also take

```
{{ Gale Crater :0.5}}
{{ Valles Marineris :0.0}}
travel complete
Success
```

10.2

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?

```
public class Oops {
  static void foo(int[] a,int n) {
    int i = a[n];
    System.out.println("i="+i);
    foo(a,i); }
  public static void main(String[] args) {
    try {
      try {
        foo(new int[]{1,2,3,4,-1},0);
        System.out.println("zero");
      } catch(Exception e) {
        foo(null,0);
        System.out.println("one");
      } finally {
        System.out.println("two"); }
    } catch(Exception e2) {
```

```
      System.out.println("three");
    } finally {
      System.out.println("four"); } } }
```

```
i=1
i=2
i=3
i=4
i=-1
two
three
four
```

10.3

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);
    }
  }
}
```

Subtracting 1.0 does not cause an exception to be thrown, so the exception handling code does not execute. 3.2

10.4

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?

```
public class Stacker {
  static void stack(int n) {
    if(n == 0) throw new RuntimeException();
    System.out.println("before="+n);
    stack(n-1);
    System.out.println("after="+n);
  }
  public static void main(String[] args) {
    Integer a = null;
    try {
      stack(2);
    } catch(RuntimeException re) {
    }
  }
}
```

```
before=2
before=1
```

10.5

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:8)
```

10.6

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);
    }
  }
}
```

Trying to subtract 10 causes an exception to be thrown. The exception is caught, and only 1.0 is subtracted. 4.2

10.7

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

```
public class Show {
  public static void main(String[] args) {
    new Exception().printStackTrace();
    --------------------
    System.err.println("Exception␣printed!");
  }
}
```

// output:

```
java.lang.Exception
        at Show.main(Show.java:3)
Exception printed!
```

10.8

When is an infinte loop not infinite? 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?

```
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);
  }
}
```

```
sum = 9
```

10.9

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?

```
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) {}
  } }
```

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

10.10

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?

```
public class Oops {
  static void foo(int[] a,int n) {
    int i = a[n];
    System.out.println("i="+i);
    foo(a,i); }
  public static void main(String[] args) {
    try {
      try {
        foo(new int[]{2,3,6,4,-1},0);
        System.out.println("zero");
      } catch(Exception e) {
        foo(null,0);
        System.out.println("one");
      } finally {
        System.out.println("two"); }
    } catch(Exception e2) {
      System.out.println("three");
    } finally {
      System.out.println("four"); } } }
```

```
i=2
i=6
two
three
four
```

10.11

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) {
      npe.printStackTrace();
      --------------------
    }
  }
}
```

// output:

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

10.12

It's so hard to say goodbye. 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?

```
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) {}
  }
}
```

```
Hello Alice
Hello Bob
```

11 File IO

11.1

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());
    } } }
```

```
The code does not compile.

replace the code:     public static void main(String[] args) {

with the code:   public static void main(String[] args) throws IOException {
```

11.2

What is java.io.File, and what is it used for?
It represents a file on disk. The File object can be used to delete a file or to check whether it exists. It can also be passed in the constructor of java.util.Scanner to read a file.

11.3

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("birds.txt");
    FileWriter fw = new FileWriter(f);
    fw.write("Holy␣Mackerel\n");
    fw.close();
  }
}
```

```
The code does not compile.

replace the code:     public static void main(String[] args) {
```

with the code: public static void main(String[] args) throws IOException {

11.4

Fill in the missing code.

```
import java.io.FileWriter;
--------------------
import java.io.IOException;
import java.util.Scanner;
public class FilesWrite {
  public static void main(String[] args) throws IOException {
    String fileName = "/tmp/out.txt";
                      new FileWriter(fileName);
    FileWriter fw = -------------------------
    fw.write("Hello");
    fw.close();
  }
}
```

11.5

Write a code that prints "Hello, World." to a file named hello.txt.

```
import java.io.*;
import java.util.*;
public class HelloFile {
  public static void main(String[] args) throws IOException {
    File f = new File("hello.txt");
    FileWriter fw = new FileWriter(f);
    fw.write("Hello, World\n");
    fw.close();
    Scanner s  = new Scanner(f);
    assert s.nextLine().trim().equals("Hello, World");
    f.delete();
  }
}
```

11.6

Formatting using String.format() or System.out.printf().
Find the best match between the column on the left and the column on the right.

(1) %d	7	(a) floating point with 2 decimal places
(2) %3s	5	(b) integer, width 4 chars, zeros in front
(3) %4d	6	(c) string
(4) %f	2	(d) string, width 3 chars
(5) %04d	4	(e) floating point
(6) %s	8	(f) new line, os independent
(7) %.2f	3	(g) integer, width 4 chars
(8) %n	1	(h) integer

11.7

Reading and Writing from files.
Find the best match between the column on the left and the column on the right.

(1) FileWriter	5	(a) write to memory
(2) BufferedWriter	7	(b) read from a file
(3) Scanner	1	(c) write to a file
(4) Writer	3	(d) parse input
(5) StringWriter	6	(e) has printf
(6) PrintWriter	2	(f) just an optimization
(7) FileReader	4	(g) superclass of FileWriter and StringWriter

11.8

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());
    } } }
```

12 Regular Expressions

12.1

Fill in the missing code.

```
    String in = "225-123-4567\n982-333-4444\n115-765-4321\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {
                      "(\\d+)-(\\d+)-(\\d+)";
      String pattern = ----------------------
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();
                      Integer.parseInt(mr.group(1))
        int areaCode = ----------------------------;
                      Integer.parseInt(mr.group(2))
        int exchange = ----------------------------;
                    Integer.parseInt(mr.group(3))
        int number = ----------------------------;
        System.out.printf("(%03d)%03d-%04d%n",areaCode,exchange,number);
      }  else { System.out.println("Incorrect");
      } s.nextLine();
```

// output:

```
(225)123-4567
(982)333-4444
(115)765-4321
```

12.2

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()) {
                        "(\\w+)\\s+(\\d+)-(\\d+)-(\\d+)"
      String pattern = ------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();
                                 mr.group(2)
        int a = Integer.parseInt(-----------);
                                 mr.group(3)
        int b = Integer.parseInt(-----------);
                                 mr.group(4)
        int c = Integer.parseInt(-----------);
                    mr.group(1)
        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
```

12.3

Fill in the missing code.

```
    String in = "Methuselah, age:938\nWilma,age: 45\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {
                        "(\\w+)\\s*,\\s*age:\\s*(\\d+)"
      String pattern = ------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();
                                 mr.group(2)
        int a = Integer.parseInt(-----------);
                    mr.group(1)
        String b = -----------;
        System.out.printf("Name: %s, age: %d%n",b,a);
      } s.nextLine();
```

// output:

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

12.4

Fill in the missing code.

```
    String in = "9+ 104 =113\n8 +4= 5\n";
    Scanner s = new Scanner(in);
    while(s.hasNextLine()) {
                        "(\\d+)\\s*\\+\\s*(\\d+)\\s*=\\s*(\\d+)"
      String pattern = ---------------------------------------;
      if(s.findInLine(pattern) != null) {
        MatchResult mr = s.match();
                                 mr.group(1)
        int a = Integer.parseInt(-----------);
                                 mr.group(2)
        int b = Integer.parseInt(-----------);
                                 mr.group(3)
        int c = Integer.parseInt(-----------);
        if(a + b == c) System.out.println("Correct");
```

```
        else System.out.println("Incorrect");
      } s.nextLine();
```

// output:

```
Correct
Incorrect
```

12.5

Regular expressions. Find the best match between the column on the left and the column on the right.

(1) *	8	(a) means a digit
(2) \w	7	(b) means a group
(3) \t	6	(c) means a whitespace character (space, tab, newline)
(4) \b	4	(d) means word boundary
(5) \n	3	(e) means tab
(6) \s	2	(f) means a letter, digit, or underscore
(7) ()	9	(g) means anything
(8) \d	5	(h) means line feed
(9) .	10	(i) means 1 or more
(10) +	1	(j) means 0 or more

12.6

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
```

13 Linked Lists

13.1

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

```
public class Linked {
    static class Node { Object data; Node next, prev; }
    Node head, tail;
    public void add(Object d) {
        Node n = new Node();
```

```
        n.data = d;
        if(head == null) {
            head = tail = n;
            --------------------
        } else {
            tail.next = n;
            --------------------
            n.prev = tail;
            --------------------
            tail = n;
            --------------------
        } } }
```

13.2

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

- They require more overhead, specifically O(N) overhead.
- It is slow to access specific element by position: O(N).

13.3

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

- They don't require a continuous memory block.
- It is cheap to insert anywhere in the list: O(N).
- It is cheap to remove anywhere in the list: O(N).

13.4

Fill in the missing code.

```
public class Link {
  String value;
  final Link next;
  public Link(String val,Link next) {
    value = val; this.next = next;
  }
  public static void main(String[] args) {
```

```
    Link l1 = new Link("a",new Link("b",new Link("c",null)));
                  l1; x != null; x = x.next
    for(Link x = -------------------------) {
      System.out.println("item␣=␣"+x.value);
    }
  }
}
```

// output:

```
item = a
item = b
item = c
```

13.5

Create a circular singly linked list. This means that the "next" field on Node should never be null. It also means if you keep following the "next" pointer, you will come back to the node you started with.
Note that instead of keeping two pointers, one for head and one for tail, we have a single tail pointer.
Fill in the missing code.

```
class Node { Node next; Object data; }
public class CircularLinkedList {
    Node tail;
                                return tail.next.data;
    public Object getHead() { ----------------------- }
    public void add(Object d) {
        Node n = new Node();
        --------------------
        n.data = d;
        --------------------
        if(tail == null) {
        --------------------
            tail = n.next = n;
            --------------------
        } else {
        --------------------
            n.next = tail.next;
            --------------------
            tail.next = n;
            --------------------
            tail = n;
            --------------------
        } }
}
```

13.6

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

```
public class Link {
  String value;
  final Link next;
  public Link(String val,Link next) {
    value = val; this.next = next;
  }
  public String toString() {
    if(next != null)
    --------------------
      return value + "," +next;
      --------------------
```

```
    else
    --------------------
      return value;
      --------------------
  }
  public static void main(String[] args) {
    Link l1 = new Link("a",new Link("b",new Link("c",null)));
    System.out.println(l1);
  }
}
```

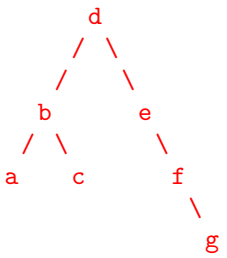
// output:

```
a,b,c
```

14 Binary Trees

14.1

The following values are added to a binary tree. Please draw the resulting tree structure. d b e f a c g



14.2

Consider the Btree below: Fix the add method. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.*;
class Node<T> {
  T data;
  Node<T> L, R;
  Node(T v) { data = v; } }
public class BTree<T extends Comparable<T>> {
  Node<T> head;
  public void add(T key) {
    if(head == null) head = new Node<>(key);
    else add(head,key); }
  private void add(Node<T> n,T key) {
    if(key.compareTo(n.data) > 0) {
    --------------------
      if(n.L == null) n.L = new Node<>(key);
      --------------------
      else add(n.L,key); }
      --------------------
    if(key.compareTo(key) < 0) {
    --------------------
      if(n.R == null) n.R = new Node<>(key);
      --------------------
      else add(n.R,key);
      --------------------
      } }
}
```

14.3

Find the best match between the column on the left and the column on the right.

(1) HashMap	4	(a) requires a Comparator or Comparable
(2) Set	6	(b) requires a hashCode
(3) TreeMap	7	(c) an Interface
(4) TreeSet	3	(d) key value pairs traversed in order
(5) ArrayList	2	(e) unique items
(6) HashSet	5	(f) efficient item lookup
(7) Queue	8	(g) a subclass of Vector
(8) Stack	9	(h) efficient insert and delete
(9) LinkedList	1	(i) key values pairs maybe in order

14.4

Which of the following values is the best hash code for String s?

1. s.charAt(0)
2. 1
3. s.length() ← This one

Answer 1 won't work if the string has zero length.
Answer 2 is usable, but highly inefficient.
Answer 3 will always work and provide some discrimination between items.

14.5

What is wrong with this code?

```
import java.util.*;
public class Monster implements Comparable<Monster> {
  int numTeeth, numClaws;
  String name;
  Monster(String n,int t,int c) { name = n; numTeeth = t; numClaws = 2; }
  public int compareTo(Monster that) {
    int diff = this.numTeeth - that.numTeeth;
    if(diff == 0) diff = this.numClaws - that.numClaws;
    return diff;
  }
}
```

The number of teeth and claws are not final, therefore sorts will be undone and binary tree lookups will fail.

14.6

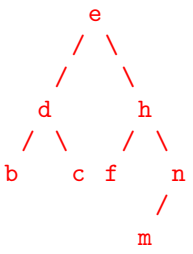
What is wrong with this code?

```
import java.util.*;
public class MathWizard {
  final int level;
  int spells, id;
  MathWizard(int id,int lvl,int spells) {
    this.id = id; this.level = lvl; this.spells = spells;
  }
  public int hashCode() { return id; }
  public static void main(String[] args) {
    Set<MathWizard> s = new TreeSet<>();
    s.add(new MathWizard(1,2,3));
  }
}
```

MathWizard is not Comparable, which is what would be required for TreeSet. Alternatively, Set s could be initialized as a hash set. As it is, this code fails with a ClassCastException.

14.7

The following values are added to a binary tree. Please draw the resulting tree structure. e d b c h f n m



14.8

The following binary tree implementation needs to be fleshed out.
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?

```
class Node {
  Comparable item;
  Node L, R;
  Node(Comparable c) { item = c; }
}
public class BTree {
  Node head;
  public void add(Comparable item) {
    if(head == null) head = new Node(item);
    else add(head,item);
  }
  private static void add(Node n,Comparable item) {
    int d=item.compareTo(n.item);
    if(d < 0) {
      // add left
      add(n.L,item);
    } else if(d > 0) {
      // add right
      add(n.R,item);
    }
}
```

The code compiles, but when it runs it throws a NullPointerException

replace the code: // add left

with the code: if(n.L == null) n.L = new Node(item); else

replace the code: // add right

with the code: if(n.R == null) n.R = new Node(item); else

14.9

The following binary tree implementation needs to be fleshed out and made generic. Do both things.
Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
class Node {
  Comparable item;
  Node L, R;
```

```
  Node(Comparable c) { item = c; }
}
public class BTree {
  Node head;
  public void add(Comparable item) {
    if(head == null) head = new Node(item);
    else add(head,item);
  }
  private static void add(Node n,Comparable item) {
    int d=item.compareTo(n.item);
    if(d < 0) {
    --------------------
      if(n.L == null) n.L = new Node(item);
      --------------------
      else add(n.L,item);
      --------------------
    } else if(d > 0) {
    --------------------
      if(n.R == null) n.R = new Node(item);
      --------------------
      else add(n.R,item);
      --------------------
    }
    --------------------
}

class Node<T> {
  T item;
  Node L, R;
  Node(T c) { item = c; }
}
public class BTree<T extends Comparable<T>> {
  Node<T> head;
  public void add(T item) {
    if(head == null) head = new Node<>(item);
    else add(head,item);
  }
  private static <T extends Comparable<T>> void add(Node<T> n,T item) {
    int d=item.compareTo(n.item);
    if(d < 0) {
      if(n.L == null) n.L = new Node<>(item);
      else add(n.L,item);
    } else if(d > 0) {
      if(n.R == null) n.R = new Node<>(item);
      else add(n.R,item);
    }
}
```

14.10

Consider the Btree below: Fix the add method. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.*;
class Node {
  int data;
  Node L, R;
  Node(int v) { data = v; } }
public class BTree {
  Node head;
```

```
  public void add(int key) {
    if(head == null) head = new Node(key);
    else add(head,key); }
  private void add(Node n,int key) {
    if(key > n.data) {
      if(n.L == null) n.L = new Node(key);
      --------------------
      else add(n.L,key);
    }
    if(key < n.data) {
      if(n.R == null) n.R = new Node(key);
      --------------------
      else add(n.R,key);
    } }
  void fill(List<Integer> list) { fill(list,head); }
  void fill(List<Integer> list,Node n) {
    fill(list,n.L);
    list.add(n.data);
    fill(list,n.R); }
  public static void main(String[] args) {
    BTree b = new BTree();
    for(int i : new int[]{1,3,8,2,9,4,7,6,16}) b.add(i);
    List<Integer> list = new ArrayList<>();
    b.fill(list);
    System.out.println(list); }
}
```

14.11

What does the fill method do in the code above? What is wrong with it? How can you fix it? Add the code "if(n == 0) return"; to the beginning of the "fill(List¡Integer¿ list,Node n)" method.

14.12

What is the output of this code? [16 , 9 , 8 , 7 , 6 , 4 , 3 , 2 , 1]

14.13

Consider the Btree below: Fix the add method. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.*;
class Node {
  int data;
  Node L, R;
  Node(int v) { data = v; } }
public class BTree {
  Node head;
  public void add(int key) {
    if(head == null) head = new Node(key);
    else add(head,new Node(key)); }
  private void add(Node d,Node k) {
    if(k.data > d.data) {
      if(d.R == null) d.R = k;
      --------------------
      else add(d.R,k);
    }
    if(k.data < d.data) {
      if(d.L == null) d.L = k;
      --------------------
```

```
      else add(d.L,k);
    } }
  void print() { print(head); }
  void print(Node n) {
    if(n == null) return;
    System.out.print("("+n.data+")");
    print(n.R); }
  public static void main(String[] args) {
    BTree b = new BTree();
    for(int i : new int[]{1,3,8,2,9,4,7,6,16}) b.add(i);
    b.print(); }
}
```

14.14

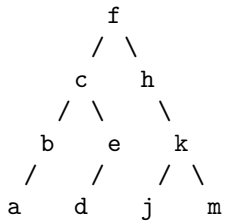
What does the fill method do in the code above? What is wrong with it? How can you fix it? Add the code "print(n.L)"; to the beginning of the "print(Node n)" method, right after the if == null check.

14.15

What is the output of this code? (1) (2) (3) (4) (6) (7) (8) (9) (16)

14.16

Consider the following binary tree.



Which of the sequences of node values below could *NOT* have produced the tree?

1. f c h b e k a d j m
2. f h c k e b m j d a
3. f c b a e d h k j m
4. f c b a d e h k j m
5. f h k m j c e d b a

f c b a d e h k j m

15 Hash Tables

15.1

Which of the following values is the best hash code for an ArrayList<Integer> alist

1. alist.get(0)
2. 666
3. alist.size() ← This one

15.2

Which of the following values is the best hash code for the Bug class:

```
public class Bug {
    int legs, arms;
    String species; }
```

1. arms + legs + species.hashCode()
2. System.identityHashCode(this)
3. super.hashCode()

The answer is "arms + legs + species.hashCode()" The other two answers are the same. Both will return the location of the object in memory. That means that even if arms, legs, and species are all the same, the hash codes might be different.

16 Generics

16.1

The following class holds the minimum value passed to it via the setIfLower() method. Make it contain a generic type instead of an Integer.

```
public class MinHolder {
    Integer data;
    public void setIfLower(Integer d) {
        if(data == null || d < data) data = d; }
    public Integer get() { return data; } }

public class MinHolder<T extends Comparable<T>> {
    T data;
    public void setIfLower(T d) {
        if(data == null || d.compareTo(data) < 0) data = d; }
    public Integer get() { return data; } }
```

16.2 Make a BTree of objects using Comparable.

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?

```
class Node<T> {
  T data;
  Node<T> L, R;
  Node(T v) { data = v; } }
public class BTree<T> {
  Node<T> head;
  public void add(T key) {
    if(head == null) head = new Node<>(key);
    else add(head,key); }
  private void add(Node<T> n,T key) {
    if(key.compareTo(n.data) > 0) {
      if(n.L == null) n.L = new Node<>(key);
      else add(n.L,key);
    }
    if(key.compareTo(n.data) < 0) {
      if(n.R == null) n.R = new Node<>(key);
      else add(n.R,key);
    } }
}
```

```
The code does not compile.

replace the code: public class BTree<T> {

with the code: public class BTree<T extends Comparable<T>> {
```

16.3

Write a generic method to count the number of elements in an array that are equal to a given argument. [1] Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
import java.util.List;
public class Count {
    public static<T> int count(List<T> list,T arg) {
    --------------------
        int n = 0;
        --------------------
        for(T t : list)
        --------------------
            if(t.equals(arg)) n++;
            --------------------
        return n;
        --------------------
    }
    --------------------
}
```

16.4

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?

```
public class Min<T> {
    public static T getMin(T a,T b) {
        if(a.compareTo(b) < 0)
            return a;
        else
            return b;
    }
}
```

```
The code does not compile.

replace the code: public static T getMin(T a,T b) {

with the code:     public static<T extends Comparable<T>> T getMin(T a,T b) {
```

16.5

What are some advantages of using generic types? [2]

1. It provides type safety
2. It reduces the need for casting

[1] from https://docs.oracle.com/javase/tutorial/java/generics/QandE/generics-questions.html
[2] from http://www.baeldung.com/java-generics-interview-questions

16.6

[3] 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.util.*;
class genericstack <E> {
    Stack <E> stk = new Stack <E>();
    public void push(E obj)
    {
        stk.push(obj);
    }
    public E pop()
    {
        E obj = stk.pop();
        return obj;
    }
}
public class Output
{
    public static void main(String args[])
    {
        genericstack <String> gs = new genericstack<String>();
        gs.push("Hello");
        System.out.println(gs.pop());
    }
}
```

Hello

16.7 Make a BTree of objects using a Comparator.

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?

```
class Node<T> {
  T data; Node<T> L, R;
  Node(T v) { data = v; } }
public class BTree<T> {
  Node<T> head; Comparator<T> c;
  public BTree(Comparator<T> c) { this.c = c; }
  public void add(T key) {
    if(head == null) head = new Node<>(key);
    else add(head,key); }
  private void add(Node<T> n,T key) {
    int diff = key.compareTo(n.data);
    if(diff > 0) {
      if(n.L == null) n.L = new Node<>(key);
      else add(n.L,key);
    } else if(diff < 0) {
      if(n.R == null) n.R = new Node<>(key);
      else add(n.R,key);
    } }
}
```

The code does not compile.

replace the code: int diff = key.compareTo(n.data);

[3] From https://www.sanfoundry.com/java-mcqs-generics/

with the code: int diff = c.compare(key,n.data);

16.8 Sort using a Comparator.

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?

```
public class Sorter<T> {
  Comparator<T> c;
  public Sorter(Comparator<T> c) { this.c = c; }
  void sort(List<T> list) {
    for(int i=0;i<list.size();i++) {
      for(int j=i+1;j<list.size();j++) {
        list.get(i).compareTo(list.get(j));
        if(diff > 0) {
          T tmp = list.get(i);
          list.set(i, list.get(j) );
          list.set(j, tmp );
        } } } }
}
```

The code does not compile.

replace the code: list.get(i).compareTo(list.get(j));

with the code: int diff = c.compare(list.get(i),list.get(j));

16.9

The following code dies with a ClassCastException on the line indicated whenever it is used to permute a list with more than one element. What's wrong? How would generics have solved the problem?

```
import java.util.ArrayList;

public class Permute {
    // Return a list of lists, where each list in the
    // return list is a different permutation of the input list.
    public static ArrayList permutations(ArrayList list) {
        ArrayList permuted_list = new ArrayList();

        if(list.size() < 2)
          return list;

        for(int i=0;i<list.size();i++) {
            ArrayList shorter_list = new ArrayList();
            for(int j=0;j<list.size();j++) {
                if(i != j)
                    shorter_list.add(list.get(j));
            }
            for(Object o : permutations(shorter_list)) {
                // ClassCastException occurs on this line
                ArrayList ilist = (ArrayList)o;
                ilist.add(list.get(i));
                permuted_list.add(ilist);
            }
        }
        return permuted_list;
```

```
    }
}
```

The problem is what happens when the list size is less than 2. The code above returns a list instead of a list of lists. Generic code prevents the problem by producing a compile error when the programmer attempts to return a list of the wrong type.

16.10

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.util.*;
public class Coord implements Comparable<Coord> {
  final int x,y;
  public Coord(int x,int y) { this.x = x; this.y = y; }
  @Override
    public int compareTo(Object o) { Coord that = (Coord)o;
    return 10000*(this.x - that.x) + (this.y - that.y);
  }
  public static void main(String[] args) {
    List<Coord> li = new ArrayList<>();
    li.add(new Coord(3,4));
    li.add(new Coord(9,2));
    Collections.sort(li);
  }
}
```

The code does not compile.

replace the code: public int compareTo(Object o) { Coord that = (Coord)o;

with the code: public int compareTo(Coord that) {

16.11

Implement the following comparable interface and use generics. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class RaceCar implements Comparable {
  public int compareTo(Object o) {
    RaceCar that = (RaceCar)o;
    --------------------
    int diff = 0;
    --------------------
    if(this.topSpeed < that.topSpeed) diff = 1;
    --------------------
    else if(this.topSpeed > that.topSpeed) diff = -1;
    --------------------
    if(diff == 0) diff = this.number - that.number;
    --------------------
    return diff;
    --------------------
  }
  double topSpeed; // sort by this first
  int number; // sort by this if num_legs is equal
}

public class RaceCar implements Comparable<RaceCar> {
  public int compareTo(RaceCar that) { ... } }
```

16.12 Sort Comparable objects.

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?

```
public class Sorter<T> {
  void sort(List<T> list) {
    for(int i=0;i<list.size();i++) {
      for(int j=i+1;j<list.size();j++) {
        int diff = list.get(i).compareTo(list.get(j));
        if(diff > 0) {
          T tmp = list.get(i);
          list.set(i, list.get(j) );
          list.set(j, tmp );
        } } } }
}
```

The code does not compile.

replace the code: public class Sorter<T> {

with the code: public class Sorter<T extends Comparable<T>> {

16.13

What is type erasure? [4]
It means that generic types are only available at compile time, not run time.

16.14

This question will not be on the exam.[5] 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.util.*;
class Animal {}
class Cat extends Animal {}
public class Farm {
    List<Animal> animals = new ArrayList<>();
    public void addAnimals(List<Animal> moreAnimals) {
        animals.addAll(moreAnimals);
    }
    public static void main(String[] args) {
        List<Cat> cats = new ArrayList<>();
        Farm f = new Farm();
        f.addAnimals(cats);
    }
}
```

The code does not compile.

replace the code: public void addAnimals(List<Animal> moreAnimals) {

with the code: public void addAnimals(List<? extends Animal> moreAnimals) {

[4] from http://www.baeldung.com/java-generics-interview-questions
[5] from http://www.baeldung.com/java-generics-interview-questions

16.15

What if a generic type is omitted when instantiating an object? Will the code compile? [6]
Yes, but there will likely be a warning from the compiler. Omitting the generic type is bad practice.

[6] from `http://www.baeldung.com/java-generics-interview-questions`