

1. [10] Each of the following needs to be inserted in the code alongside. There is only one box where it makes sense. Write the appropriate letter in the box within the code (I did one of them for you).

- (A) `super(who);`
 (B) `super(name, "Republican");`
 (C) `super(name);`
 (D) `this.party = party;`
 (E) `super(name, "Democratic");`

2. [4] After you correctly answer question 1 and try to compile the code you get the error message

Republican is abstract;
cannot be instantiated

Strike through the line of code alongside that causes that error message.

3. [6] After you've struck out that one bad line, the code will compile and when run it will print out three lines of code. One of those lines is "Hillary and Barack are wonderful! Vote for Bugs Bunny the Democratic". What are the other two lines of output.

```
import java.util.*;
abstract class Politician {
    protected String myName, party;
    public Politician(String name, String party) {
        myName = name;
        
    }
    public abstract String speak();
    public String campaign() {
        return "Vote for " + myName + " the " + party;
    }
}
class Democrat extends Politician {
    public Democrat(String name) {
        
    }
    public String speak() {
        return "Hillary and Barack are wonderful!";
    }
}
abstract class Republican extends Politician {
    public Republican(String name) {
        
    }
}
class NeoCon extends Republican {
    public NeoCon(String name) {
        
    }
    public String speak() {
        return "George and Dick are wonderful!";
    }
}
class FiscalConservative extends Republican {
    public FiscalConservative(String who) {
        
    }
    public String speak() {
        return "Budget deficits are bad!";
    }
    public String campaign() {
        return "Vote for sound financial management!";
    }
}
public class Exam1Question1 {
    Vector<Politician> crooks;
    public Exam1Question1() {
        crooks = new Vector<Politician>();
        crooks.add(new Republican("Elmer Fudd"));
        crooks.add(new FiscalConservative("Porky Pig"));
        crooks.add(new NeoCon("Yosemite Sam"));
        crooks.add(new Democrat("Bugs Bunny"));
    }
    public void speak() {
        for (Politician p : crooks) {
            System.out.println(p.speak() + " " + p.campaign());
        }
    }
    public static void main(String[] args) {
        Exam1Question1 q1 = new Exam1Question1();
        q1.speak();
    }
}
```

4. [20] Here are extracts from the code we used to implement `ArraySet<T>`

```
import java.util.*;

public class ArraySet<T> implements SetADT<T>, Iterable<T>
{
    private static Random rand = new Random();
    private final int DEFAULT_CAPACITY = 100;
    private final int NOT_FOUND = -1;
    private int count; // the current number of elements in the set
    private T[] contents;

    /**
     * Creates an empty set using the default capacity.
     */
    public ArraySet()
    {
        count = 0;
        contents = (T[])(new Object[DEFAULT_CAPACITY]);
    }

    /**
     * Returns true if this set contains the specified target
     * element.
     */
    public boolean contains (T target)
    {
        int search = NOT_FOUND;

        for (int index=0; index < count && search == NOT_FOUND; index++)
            if (contents[index].equals(target))
                search = index;

        return (search != NOT_FOUND);
    }

    /**
     * Returns true if this set is empty and false otherwise.
     */
    public boolean isEmpty()
    {
        return (count == 0);
    }

    /**
     * Adds the specified element to the set if it is not already
     * present. Expands the capacity of the set array if necessary.
     */
}
```

```

*****/
public void add (T element)
{
    if (!(contains(element)))
    {
        if (size() == contents.length)
            expandCapacity();

        contents[count] = element;
        count++;
    }
}
/*****
    Removes the specified element from the set and returns it.
    Throws an EmptySetException if the set is empty and a
    NoSuchElementException if the target is not in the set.
*****/
public T remove (T target) throws EmptySetException,
                NoSuchElementException
{
    int search = NOT_FOUND;

    if (isEmpty())
        throw new EmptySetException();

    for (int index=0; index < count && search == NOT_FOUND; index++)
        if (contents[index].equals(target))
            search = index;

    if (search == NOT_FOUND)
        throw new NoSuchElementException();

    T result = contents[search];

    contents[search] = contents[count-1];
    contents[count-1] = null;
    count--;

    return result;
}
/*****
    Removes a random element from the set and returns it. Throws
    an EmptySetException if the set is empty.
*****/
public T removeRandom() throws EmptySetException
{
    if (isEmpty())

```

```
        throw new EmptySetException();

    int choice = rand.nextInt(count);

    T result = contents[choice];

    contents[choice] = contents[count-1]; // fill the gap
    contents[count-1] = null;
    count--;

    return result;
}
```

(a) At a stage when the `ArraySet` contains n elements, what is the big O runtime for the `add(T o)` method?

(b) At a stage when the `ArraySet` contains n elements, what is the big O runtime for the `remove(T o)` method?

(c) At a stage when the `ArraySet` contains n elements, what is the big O runtime for the `removeRandom()` method?

(d) Draw a picture of the contents of the array after you have executed

```
ArraySet<String> people = new ArraySet<String>();  
people.add("Tom");  
people.add("Dick");  
people.add("Harry");
```

(e) What is the output of:

```
ArraySet<String> people = new ArraySet<String>();  
people.add("Tom");  
people.add("Dick");  
people.add("Harry");  
System.out.println(people.toString());  
try {  
    people.remove("Dick");  
}  
catch(Exception e) {  
}  
people.add("Alice");  
System.out.println(people.toString());  
}
```

5. [20] Here are extracts from the code we used to implement RPJLinkedSet<T>

```
public class RPJLinkedSet<T> implements SetADT<T>
{
    // instance variables
    private ANode<T> contents;
    private int count;

    /**
     * Constructor for objects of class RPJLinkedSet
     */
    public RPJLinkedSet()
    {
        contents = new ENode<T>();
        count = 0;
    }

    /**
     * add
     * Adds one element to this set, ignoring duplicates
     *
     * @param element
     */
    public void add (T element) {
        if (!contains(element)) {
            contents = contents.insert(element);
            count++;
        }
    }

    /**
     * Removes and returns a random element from this set.
     */
    public T removeRandom () throws EmptySetException {
        int place = (int)(Math.random() * size());
        if (place == 0) {
            T ret = contents.getData();
            contents = contents.getNext();
            count--;
            return ret;
        }
        ANode<T> finger = contents;
        place = place - 1;
        while (place > 0) {
            finger = finger.getNext();
            place--;
        }
    }
}
```

```

    }
    T ret = finger.getData();
    finger.setNext((finger.getNext()).getNext());
    count--;
    return ret;
}

/** Removes and returns the specified element from this set. */
public T remove (T element) throws EmptySetException, NoSuchElementException {
    ANode<T> finger = contents.find(element);
    try {
        finger.setData(finger.getNext().getData()); // copy data from next node
        finger.setNext(finger.getNext().getNext()); // short-circuit next node
    }
    catch (EmptySetException e) {
        // This must have been the last node in the list, so
        contents = contents.removeLast();
    }
    count--;
    return element;
}

/** Returns true if this set contains the parameter */
public boolean contains (T target) {
    return contents.contains(target);
}

/** Returns true if this set contains no elements */
public boolean isEmpty() {
    return contents.isEmpty();
}
}

```

Additionally, I am providing you with the following big O information for the ANode / Node / ENode class hierarchy

- All constructors are $O(1)$
- `contains(T element)` is $O(n)$
- `isEmpty()` is $O(1)$
- `removeLast()` is $O(n)$

Finally, here is the code from the ANode class for `insert(T element)`:

```

public ANode<T> insert(T data) {
    return new Node<T>(data, this);
}

```

(a) At a stage when the `RPJLinkedSet` contains n elements, what is the big O runtime for the `add(T o)` method?

(b) At a stage when the `RPJLinkedSet` contains n elements, what is the big O runtime for the `remove(T o)` method?

(c) At a stage when the `RPJLinkedSet` contains n elements, what is the big O runtime for the `removeRandom()` method?

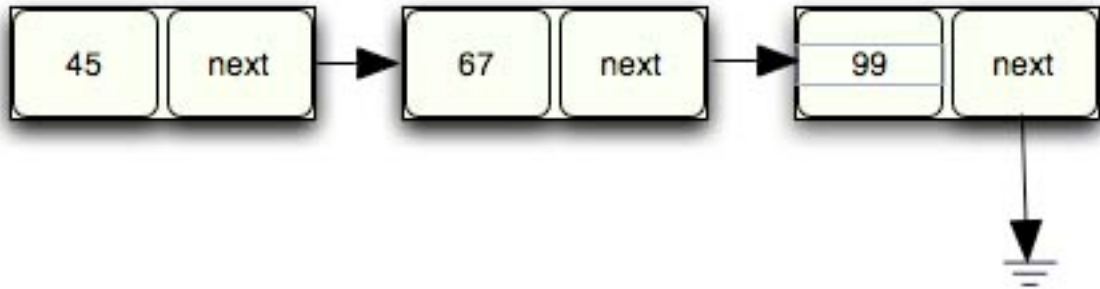
(d) Draw a picture of the contents of the linked list after you have executed

```
RPJLinkedSet<String> people = new RPJLinkedSet<String>();
people.add("Tom");
people.add("Dick");
people.add("Harry");
```

(e) What is the output of:

```
RPJLinkedSet<String> people = new RPJLinkedSet<String>();
people.add("Tom");
people.add("Dick");
people.add("Harry");
System.out.println(people.toString());
try {
    people.remove("Dick");
}
catch(Exception e) {
}
people.add("Alice");
System.out.println(people.toString());
}
```

6. [20] Recall that the implementation of Ordered set is very similar to RPJLinkedSet except that we ensure that the items stored in an OrderedSet appear in their natural order when you traverse the underlying structure. So that if the integers 67, 99, and 45 are added to an initially empty `OrderedSet<Integer>` the underlying structure looks like



I won't give you any code for ordered sets, but try to recall the basic idea behind the ordered set implementation we developed in class. Operations were developed to ensure that the nodes of the underlying linked list were ordered according to their data contents, as in the diagram above.

(a) At a stage when the `OrderedSet` contains n elements, what is the big O runtime for the `add(T o)` method?

(b) At a stage when the `OrderedSet` contains n elements, what is the big O runtime for the `remove(T o)` method?

(c) At a stage when the `OrderedSet` contains n elements, what is the big O runtime for the `removeRandom()` method?

(d) Draw a picture of the contents of the underlying linked list after you have executed

```
OrderedSet<String> people = new OrderedSet<String>();
people.add("Tom");
people.add("Dick");
people.add("Harry");
```

(e) What is the output of:

```
OrderedSet<String> people = new OrderedSet<String>();
people.add("Tom");
people.add("Dick");
people.add("Harry");
System.out.println(people.toString());
try {
    people.remove("Dick");
}
catch(Exception e) {
}
people.add("Alice");
System.out.println(people.toString());
}
```

7.

Consider the following code:

```
public class Exam1Question7 {
    private Integer[] myArray;
    private int size;
    public Exam1Question7(int n) {
        myArray = new Integer[n];
        size = n;
        for (int i=0; i<n; i++)
            myArray[i] = (int) (1000 * Math.random());
    }
    public int mystery1() {
        int total = 0;
        for (int i=0; i<size; i++)
            total = total + myArray[i];
        return total;
    }
    public int mystery2() {
        int total = 0;
        total = size;
        return total;
    }
    public int mystery3() {
        int smallest = myArray[0];
        for (int i=1; i<size; i++)
            if (myArray[i] < smallest)
                smallest = myArray[i];
        return smallest;
    }
    public int mystery4() {
        int largest = myArray[0];
        for (int i=1; i<size; i++)
            if (myArray[i] > largest)
                largest = myArray[i];
        return largest;
    }
    public int mystery5() {
        return mystery3() + mystery4();
    }
}
```

- (a) [4] Explain in English what method `mystery1` does.

What is the big O runtime of `mystery1`?

- (b) [4] Explain in English what method `mystery2` does.

What is the big O runtime of `mystery2`?

- (c) [4] Explain in English what method `mystery3` does.

What is the big O runtime of `mystery3`?

- (d) [4] Explain in English what method `mystery4` does.

What is the big O runtime of `mystery4`?

- (e) [4] Explain in English what method `mystery5` does.

What is the big O runtime of `mystery5`?