

S O L U T I O N S
 COSC 2006 MIDTERM EXAM
 DATA STRUCTURES I

Tuesday, October 19, 2010, 10:00 am

Time Allowed: 75 minues

Instructor: Barry G. Adams

Name (PLEASE PRINT) _____

Student # _____

1. Answer ALL questions. Write your answers on this questionnaire.
 2. Use back of exam pages for rough work if necessary.
 3. Do not write comments in your programs.
 4. No aids permitted
 5. Number of Questions: 4
 6. Total Marks: 20
-

Question 1 (3 + 3 marks)

The DynamicBag<E> class implements the Bag<E> interface (see page 6) and is given by

```
public class DynamicBag<E> implements Bag<E>
{
    private E[] data;
    private int size;
    public DynamicBag(int initialCapacity) {...}
    public DynamicBag() {...}
    public DynamicBag(DynamicBag<E> b) {...}

    // interface methods are implemented here.
}
```

- (a) Write the contains method with prototype

```
public boolean contains(E element)
```

that returns true if element is in the bag else returns false.

Answer:

```
| public boolean contains(E element)
| {
|     for (int k = 0; k < size; k++)
|     {
|         if (data[k].equals(element))
|             return true;
|     }
|     return false;
| }
```

(b) Write a `remove` method with prototype

```
public boolean remove(E element)
```

that removes the given `element` from the bag and returns true if `remove` was successful else returns false.

Answer:

```
| public boolean remove(E element)
| {
|     for (int k = 0; k < size; k++)
|     {
|         if (data[k].equals(element))
|         {
|             data[k] = data[size-1];
|             data[size-1] = null;
|             size--;
|             return true;
|         }
|     }
|     return false;
| }
```

Question 2 (2 + 2 + 2 marks)

The `DynamicArray<E>` class implements the `Array<E>` interface (see page 6) and is given by

```
public class DynamicArray<E> implements Array<E>
{
    private E[] data;
    private int size;

    public DynamicArray(int initialCapacity) {...}
    public DynamicArray() {...}
    public DynamicArray(DynamicArray<E> b) {...}

    // interface methods are implemented here.
}
```

(a) Write a private `resize` method with prototype

```
private void resize()
```

that expands the array referenced by `data` to twice its size and copies the existing elements to the new array.

Answer:

```
| private void resize()
| {
|     int newCapacity = 2 * data.length;
|     E[] newData = (E[]) new Object[newCapacity];
|     for (int k = 0; k < data.length; k++) // System.arraycopy(
|     { // data, 0,
|         newData[k] = data[k]; // newData, 0,
|     } // data.length);
|     data = newData;
| }
```

(b) Write the add method with prototype

```
public boolean add(E element)
```

that adds the given `element` to the end of the array if there is room. Otherwise the array is resized. The return value is always true.

Answer:

```
| public boolean add(E element)
| {
|     if (size == data.length)
|     {
|         resize();
|     }
|     data[size] = element;
|     size++;
|     return true;
| }
```

(c) Write the add method with prototype

```
public void add(int k, E element);
```

that adds the given `element` at position `k`, by shifting elements to open a space.

Answer:

```
| public void add(int k, E element)
| {
|     if (k < 0 || k > size)
|     {
|         throw new IndexOutOfBoundsException("add: index out of bounds");
|     }
|     if (size == data.length)
|     {
|         resize();
|     }
|     // Shift element to make room for new one.
|     // This works if element is added at end (k = size())
|     for (int j = size-1; j >= k; j--)
|     {
|         data[j+1] = data[j];
|     }
|     data[k] = element; // add the element
|     size++;
| }
```

Question 3 (2 + 2 marks)

- (a) Using the `nextInt(int n)` method in the `Random` class which generates random integers in the range $0, 1, \dots, n-1$, and the `TreeSet` class that implements the `Set` interface on page 6, write some statements to create a set containing 6 distinct numbers each in the range 1 to 49. For example, $\{2, 17, 23, 35, 41, 45\}$.

Answer:

```
| Set<Integer> set = new TreeSet<Integer>();
| while (set.size() < 6)
| {
|     set.add(random.nextInt(49) + 1);
| }
```

- (b) Create a `HashSet`, `s`, of strings that implements the `Set<String>` interface. Add 4 strings to the set. Use the `iterator()` method to obtain an iterator (`hasNext()` and `next()`) and use it to traverse the set and display its elements one per line.

Answer:

```
| Set<String> s = new HashSet<String>();
| s.add("Fred");
| s.add("Gord");
| s.add("Mary");
| s.add("Janet");
| Iterator<String> iter = s.iterator();
| while (iter.hasNext())
| {
|     String str = iter.next();
|     System.out.println(str);
| }
```

Question 4 (2 + 2 marks)

Using the `HashMap` class that implements the `Map` interface on page 6

- (a) write some statements that create a name-age map of four people.

Answer:

```
| Map<String,Integer> age = new HashMap<String,Integer>();  
| age.put("Fred", 10);  
| age.put("Gord", 12);  
| age.put("Mary", 14);  
| age.put("Carol",10);
```

- (b) Write some statements that add 1 year to each age in the map using the “for each” loop. (Hint: use `keySet()` to get the set of keys to traverse.)

Answer:

```
| Set<String> keys = age.keySet();  
| for (String name : keys)  
| {  
|     int a = age.get(name);  
|     age.put(name, a + 1);  
| }
```

Interface summaries

The Bag interface

```
public interface Bag<E>
{
    int size();
    boolean isEmpty();
    boolean add(E element);
    boolean remove(E element);
    boolean contains(E element);
}
```

The Array interface

```
public interface Array<E>
{
    int size();
    boolean isEmpty();
    boolean add(E element);
    E get(int index);
    void set(int index, E element);
    void add(int k, E element);
    E remove(int k);
}
```

The Set interface

```
public interface Set<E>
{
    int size();
    boolean isEmpty();
    boolean add(E element);
    boolean contains(Object element)
    Iterator<E> iterator();
    boolean remove(Object obj);
}
```

The Map interface

```
public interface Map<K,V>
{
    int size();
    boolean isEmpty();
    boolean containsKey(Object key);
    V get(Object key);
    V put(K key, V value); // put new key value pair into map
    Set<K> keySet(); // return keys as a set
    // ... Other methods not needed
}
```

Useful result

```
System.arraycopy(Object source, int sourceIndex, Object dest, int destIndex,
    int n);
```

Here `source` is a reference to the source array, `sourceIndex` is the starting index in the source array, `dest` is a reference to the destination array, `destIndex` is the starting index in the destination array, and `n` is the number of array elements to copy.