

Initials: .....

ID Number: .....

Do **not** write your full name on the test.

## COMP103: Test

4 Sept, 2002.

### Instructions

- Time: **2 hours**.
- Answer **all** the questions.
- There are 120 marks in total.
- Write your answers in the boxes in this test paper and hand in all sheets.
- Every box with a heavy outline requires an answer.
- If you think some question is unclear, ask for clarification.

### Questions

### Marks

- |                              |      |
|------------------------------|------|
| 1. Collection Types          | [12] |
| 2. “Big O” analysis          | [12] |
| 3. Implementing ArrayBag     | [22] |
| 4. Linked Lists              | [25] |
| 5. Using Vectors             | [16] |
| 6. Hash Tables               | [18] |
| 7. Binary Search and Sorting | [15] |

Note:

This test was considered fairly difficult. Also, in 2002 we covered slightly different material from this year. However, the questions will still be good practice for the 2003 test.

**Question 1. Collection Types**

[12 marks]

(a) [4 marks] Which of the following methods must be provided by a class that implements the `Bag` interface from the `jds` library? Mark the “yes” box or the “no” box for each method.

<code>public int <u>size</u>();</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>addElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public Object <u>elementAt</u>(int i);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public boolean <u>containsElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>setElementAt</u> (Object value, int index);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>removeElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>set</u> (Object key, Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public Enumeration <u>elements</u>();</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>

(b) [4 marks] Which of the following methods must be provided by a class that implements the `Indexed` interface from the `jds` library? Mark the “yes” box or the “no” box for each method.

<code>public int <u>size</u>();</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>addElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public Object <u>elementAt</u>(int i);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public boolean <u>containsElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>setElementAt</u> (Object value, int index);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>removeElement</u>(Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public void <u>set</u> (Object key, Object value);</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>
<code>public Enumeration <u>elements</u>();</code>	yes: <input type="checkbox"/>	no: <input type="checkbox"/>

(c) [4 marks] Consider the three collection types `Bag`, `Set`, and `Indexed`. For each type, state whether it allows duplicates or not and what constraints it places on the ordering of the items.

**Question 2. Asymptotic or “Big O” analysis**

[12 marks]

**(a)** [3 marks] What are the average case asymptotic costs (“Big O”) of the following sorting algorithms?

- Bubble Sort
- Insertion Sort
- Merge Sort
- Quicksort
- Selection Sort


**(b)** [3 marks] What are the average case asymptotic costs of searching for an item in each of the following implementations of **Bag**? Assume that the size of the bag is  $n$ .

- ArrayBag (unordered, array)
- SortedVector (ordered, array)
- DoubleListBag (unordered, linked list)
- BucketHashtable (Hashtable, each bucket is an ArrayBag)
- QuadHashtable (Hashtable with open addressing, quadratic probing, and guaranteed less than 80% full).


**(c)** [3 marks] Suppose a program uses an algorithm with an average case asymptotic cost of  $O(n^2)$ . When the program is run on a case where  $n = 1,000$ , the measured running time of the program is 1.1 seconds. Give a reasonable estimate of the running time of the program on a case where  $n = 4,000$ .

--

**(d)** [3 marks] Explain why your estimate could be too high.

--

**Question 3. Implementing ArrayBag**

[22 marks]

The implementation of `ArrayBag` given in the lectures used an array to store the items in the collection. If the array is full, `addElement` creates a new array, double the size of the old one, and copies all the items over.

```
public void addElement (Object item) {
    if (elementCount == elementData.length){
        Object [ ] newArray = new Object[elementData.length*2];
        for (int i = 0; i < elementCount; i++)
            newArray[ i ] = elementData[ i ];
        elementData = newArray;
    }
    elementData[elementCount] = item;
    elementCount++;
}
```

(a) [4 marks] Explain why the average cost of adding an item to an `ArrayBag` containing  $n$  items is  $O(1)$ , even though it will take  $n$  steps whenever the array is full.

(b) [4 marks] If an `ArrayBag` contains  $n$  items and `removeElement` has never been called, what is the largest possible amount of “wasted” space (array locations not currently used to hold items) that might be in the array?

(c) [3 marks] The `removeElement` method does not reduce the size of the array when it removes items from the collection. If an `ArrayBag` contains  $n$  items, and `removeElement` has been called exactly  $m$  times, what is the largest possible amount of “wasted” space in the array?

(Question 3 continued on next page)

**(Question 3 continued)**

(d) [7 marks] The `ArrayBag` would use space more efficiently if `removeElement` checked the number of elements and replaced the array by an array of half the size whenever the array became only half full. Complete the following implementation of `removeElement` so that it does this.

```

public void removeElement (Object item) {
    for (int i = 0; i < elementCount; i++){
        if (item.equals(elementData[ i ])){
            elementCount--;
            elementData[i] = elementData[elementCount];

            return;
        }
    }
    throw new NoSuchElementException();
}

```

(e) [2 marks] Explain why this particular design is a bad idea, by describing a sequence of `addElement` and `removeElement` operations that would be unnecessarily slow.

(f) [2 marks] Suggest a modification to your version of `removeElement` that would be more time efficient and still be reasonably space efficient.

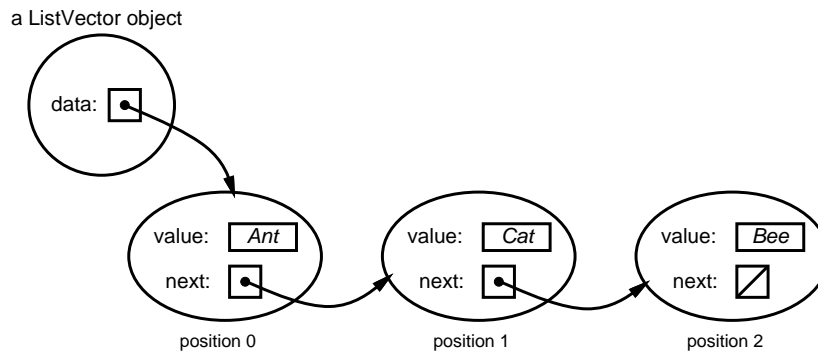
**Question 4. Linked Lists**

[25 marks]

The `Indexed` collection type describes collections of items in which the position of items in the collection is significant. The `Vector` class is an implementation of the `Indexed` type that uses an array to store the items. This question concerns the `ListVector` class that also implements the `Indexed` type, but uses a linked list data structure to store the items.

The `ListVector` class uses a singly linked list with no tail pointer. Part of the implementation is given below, along with a diagram of a `ListVector` value containing three items.

```
public class ListVector implements Indexed {
    private Node data;
    :
    private class Node{
        public Object value;
        public Node next;
        public Node(Object val, Node n){
            value = val;
            next = n;
        }
    }
}
```



You are to complete several of the method definitions for `ListVector` class.

(a) [4 marks] Complete the following definition of the `addFirst` method that inserts a new item into the collection at position 0 (the front of the list).

```
public void addFirst (Object val){
}
}
```

(Question 4 continued on next page)

**(Question 4 continued)**

**(b)** [5 marks] Complete the following definition of the `size` method that returns the number of items in the collection.

```
public int size() {
```

```
}
```

**(c)** [8 marks] Complete the following definition of the `elementAt` method that returns the item at a given position in the collection. It throws a `NoSuchElementException` if there is no such position in the collection. Note that the first element in the collection is at position 0.

```
public Object elementAt (int index) {
```

```
}
```

(Question 4 continued on next page)

**(Question 4 continued)**

(d) [8 marks] Complete the following definition of the `removeElementAt` method that removes the item at a given position in the collection, making the collection one item smaller. You may assume that the given position exists (ie, that  $0 \leq \text{position} < \text{size}()$ ).

```
public void removeElementAt (int index){
```

```
    // assumes  $0 \leq \text{index} < \text{size}()$ 
```

```
}
```

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.

**Question 5. Code with Vectors**

[16 marks]

The following `countDoubles` method is supposed to find the number of items that occur exactly twice in a `Vector`. For example, when called on a vector containing the 11 items

“s”, “t”, “e”, “g”, “a”, “s”, “a”, “u”, “r”, “u”, “s”

`countDoubles` should return 2, because “a” and “u” occur exactly twice. (“s” occurs three times and should not be counted.) This version of `countDoubles` does not work correctly on all inputs.

```
public int countDoubles(Vector v){
    // Assumes that v does not contain any null items
    int numDoubles = 0;
    for (int i = 0; i < v.size(); i++){
        int count = 1;
        for (int j = i+1; j < v.size(); j++){
            if (v.elementAt(i).equals(v.elementAt(j)))
                count++;
        }
        if (count == 2)
            numDoubles++;
    }
    return numDoubles;
}
```

(a) [3 marks] Exactly how many times will the variable `count` be set to 1 if the vector `v` contains  $n$  items? (Express your answer in terms of  $n$  but do not use “Big O”.)

(b) [2 marks] What is the asymptotic cost (“Big O”) of the `countDoubles` method if the vector `v` contains  $n$  items?

(Question 5 continued on next page)

**(Question 5 continued)**

(c) [3 marks] Give an example of a vector for which `countDoubles` will return a *correct* value of 3.

(d) [3 marks] Give an example of a `Vector` for which `countDoubles` should return a value of 3, but actually returns an incorrect value. Also state what value `countDoubles` returns on your example.

(e) [5 marks] Make minor corrections to the code for `countDoubles` so that it works correctly on all vectors.

```

public int countDoubles(Vector v){
    // Assumes that v does not contain any null items
    int numDoubles = 0;
    for (int i = 0; i < v.size(); i++){
        int count = 1;
        for (int j = i+1; j < v.size(); j++){
            if (v.elementAt( i ).equals(v.elementAt( j )))
                count++;
        }
        if (count == 2)
            numDoubles++;
    }
    return numDoubles;
}

```

**Question 6. Hash Tables**

[18 marks]

The `BucketHashtable` class from the `jds` library is implemented as an array of buckets, where each bucket is a `Bag` containing the items that hashed to that bucket. The buckets may be instances of any implementation of `Bag`.

The buckets could be instances of `ArrayBag`:

```
public class BucketHashtable implements Bag {
    private Bag[] buckets;
    public BucketHashtable(int numBuckets){
        buckets = new Bag[numBuckets];
        for (int i = 0; i < numBuckets; i++)
            buckets[ i ] = new ArrayBag();
    }
    :
}
```

(a) [4 marks] Suppose we have constructed a `BucketHashtable` with  $k$  buckets and have implemented the buckets with `ArrayBags`. If the `BucketHashtable` contains  $n$  items (where  $n$  is much larger than  $k$ ), what is the average asymptotic (“Big O”) cost of searching for an item in the `BucketHashtable`? Justify your answer.

(b) [4 marks] On average, how much faster would it be to search for an item in the `BucketHashtable` than it would be to search for an item in an `ArrayBag` containing the same number of items? Justify your answer. (Assume that the total number of items is much greater than the number of buckets)

(Question 6 continued on next page)

**(Question 6 continued)**

(c) [5 marks] Since the `BucketHashtable` class implements `Bag`, it would seem that you could use `BucketHashtables` for the buckets:

```
public class BucketHashtable implements Bag {
    private Bag[ ] buckets;
    public BucketHashtable(int numBuckets){
        buckets = new Bag[numBuckets];
        for (int i = 0; i < numBuckets; i++)
            buckets[ i ] = new BucketHashtable(numBuckets);
    }
    :
}
```

Explain why this would not work.

(d) [5 marks] The `QuadHashtable` class you wrote for assignment 5 also implements `Bag`. (`QuadHashtable` used open addressing, and quadratic probing.) `QuadHashtable` had the fastest search performance of any of the implementations of `Bag` we looked at. It would be possible to use `QuadHashtables` for the buckets in `BucketHashtable`:

```
public class BucketHashtable implements Bag {
    private Bag[ ] buckets;
    public BucketHashtable(int numBuckets){
        buckets = new Bag[numBuckets];
        for (int i = 0; i < numBuckets; i++)
            buckets[ i ] = new QuadHashtable(numBuckets);
    }
    :
}
```

Explain why this version of `BucketHashtable` could be even slower for searching than the original version using `ArrayBag` for the buckets.

**Question 7. Binary Search and Sorting Algorithms**

[15 marks]

(a) [5 marks] Suppose a Vector contained the values

ant	bee	cat	dog	egg	fly	gnu	hen	jay	kit	man	owl	pig	rat	tui	yak
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Show the sequence of values from this vector that binary search would look at if it were searching for the value "egg".

(b) [5 marks] Suppose a Vector containing the following values is to be sorted using Insertion Sort.

tui	bee	gnu	ant	pig	hen	yak	kit
0	1	2	3	4	5	6	7

Show the state of the vector at the end of the outer loop of Insertion Sort after each of the first three iterations:

after 1: 

0	1	2	3	4	5	6	7

after 2: 

0	1	2	3	4	5	6	7

after 3: 

0	1	2	3	4	5	6	7

(Question 7 continued on next page)

**(Question 7 continued)**

(c) [5 marks] Suppose a Vector containing the following values is to be sorted using Merge Sort (recursive version).

tui	bee	gnu	ant	pig	hen	yak	kit
0	1	2	3	4	5	6	7

Show the state of the vector at the end of the call to Merge after each of the first three calls,

after 1: 

0	1	2	3	4	5	6	7

after 2: 

0	1	2	3	4	5	6	7

after 3: 

0	1	2	3	4	5	6	7

\*\*\*\*\*

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.