Computer Science 136

Bruce - Spring ‘04

Midterm Examination

March 10, 2004

Question	Points	Score
1	12	____
2	10	____
3	11	____
4	18	____
5	8	____
TOTAL	59	____

Your name (Please print)

____

I have neither given nor received aid on this examination.

____

(sign here)

1. In the following problem you are to design an interface and class for a data structure which represents sets of characters. As usual, no repeated elements are allowed in these sets. Thus the collection: 'a', 'e', 'i', 'o', and 'u' is a legal set, but 'a', 'e', 'a' is not. This data structure will have two methods:
   i. insert(char newChar) is a procedure which adds newChar to the set.
   ii. contains(char findChar) is a function which returns a boolean value which indicates if findChar is an element of the set.

a. Please write a legal Java `interface` for this data structure. Be sure to include preconditions and postconditions for all methods. Please name your interface `CharSetInterface`.

b. Suppose we decide to implement `CharSetInterface` by a class in which a singly-linked list holds all of the elements in order (i.e. characters whose unicode is smaller come before those with larger unicode). Please write down the definition of this class. This should be a full and legal class definition except you do not need to include any of the method bodies (just include "{...}" where the code would usually go). Do include instance variable declarations and don't forget to include a constructor which creates an empty list. You need not repeat the pre and post-conditions from the interface. Do not forget to provide qualifiers such as public and protected.

c. If CharacterSet is implemented as given in b. above, what would the worst-case time complexity be of the operation insert if the set has n elements? (Use big-O notation.)

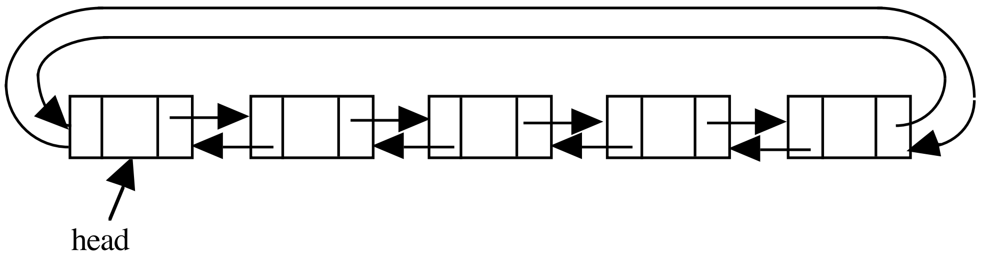
d. Suppose instead of representing the set by a linked list, we design an alternative implementation in which the set is represented by an array, rep, of booleans with subscripts ranging from 0 to 65535 (these represent the codes for all of the characters representable in unicode). For example, the unicode for 'a' is 97. Then 'a' is in the set if and only if rep[97] is true. What is the worst-case complexity of insert with this representation. (You may assume there is a constant-time function available which computes the unicode of a given letter.)

e. What might be the disadvantage of using the array representation compared to the linked list representation for this particular application.

2. A circular doubly linked list has instance variables:

```
protected DoublyLinkedListElement head;
protected int count;
```

A circular doubly linked list with four elements is represented as in the picture below:



This question revolves around the implementation of the method to remove the last element:

```
// pre: !isEmpty()
// post: returns & removes value from last elt of list
public Object removeLast()
```

a. What special case(s) must you worry about in writing this method?

b. Please write the Java code for this method. Write your answer on the next page. You may not use `removeFirst` (or any other method of `DoublyLinkedList`) in your code. Do not forget to update `count`. *(Recall that DoublyLinkedListElement has methods previous(), next(), value(), setPrevious(DoublyLinkedListElement prev), and setNext(DoublyLinkedListElement nextOne).)*

3. The following is a recursive method to remove duplicates that might be added to the DoublyLinkedList class:

```
   // post:  The list from first to end contains no
duplicates
   protected removeDups(DoublyLinkedListElement first) {
      if (first != null) {
         Object firstValue = first.value(); // value in first
         DoublyLinkedListElement finger = first.next();
         while (finger != null){
               //remove all occurrences of firstValue
            if (finger.value().equals(firstValue)){
               ... // remove finger in constant time -
omitted.
            }
            finger = finger.next();
         }
         if (first.next() == null) {
            last = first;
         } else {
            removeDups(first.next());
         }
      }
   }
```

This recursive function would normally be called by a public method like:

```
public removeAllDups(){
   removeDups(head);
}
```

a. What two things must one prove in order to prove that `removeDups` meets its postcondition? You need not give the proof itself, just state the two statements which must be proven. Be sure to state any hypotheses which are allowed to be assumed for the proof. Note: The two things you list must be statements about this particular method, not general statements about induction!

b. What is the complexity of `removeDups(first)` if there are n elements in the list from `first` to the last element? Use big-O notation. Justify your answer.

c. There are faster algorithms to remove duplicates if the elements of the lists are of type Comparable (can be compared) and we don't care whether the order of the elements is changed by the algorithm. Please describe a more efficient (in terms of big-O) algorithm to remove duplicates (a paragraph telling what must be done is fine). Provide it's big-O complexity and justify your answer.

4. Short answers:

a. Suppose we wish to add a method to our various linked list implementations to concatenate (glue together) two lists of size n and m, respectively. Executing list1.concatenate(list2) should result in changing list1 to be the concatenation of list1 and list2. The list referred to by list2 may be destroyed as a result of the operation.

What is the worst case complexity *(use big-O notation)* of the best algorithm to accomplish this for the following implementations of the list. *Please explain very briefly your answer for each*. [Note: A method that takes a parameter of the same type as the class has access to the instance variables of the parameter.]

i. Both are singly-linked with references only to the heads of the lists:

ii. Both are circular singly-linked with references only to the tails of the lists:

iii. Both are doubly-linked with references to both the heads and tails of the lists:

b. Consider the (non-circular) singly-linked and doubly-linked list and vector implementations of the List interface that were discussed in the text and class. Please provide the <u>worst-case</u> complexity of the following operations. Express your answer using big-"O" notation.

Complexity	Singly-linked list	Doubly-linked list	Vector
addFirst			
contains			
removeLast			

c. It was indicated in class that one can think of pre-conditions and post-conditions as a contract between the supplier of the method and the user (caller) of the method. Please explain briefly this analogy.

d. What are the worst-case and average-case times to insert an element at the end of a Vector? Please explain why there is a difference between the two. (Assume that the implementation uses the doubling strategy.)

5a. Please write the worst-case time complexity of the following sorts. The options are: O(1), O(log n), O(n), O(n log n), O(n^2), O(n^3), O(2^n).

i. Insertion sort

ii. Selection sort:

iii. Merge sort

b. Describe an important case in which an insertion sort might be more appropriate (and efficient) than a merge sort. You may assume the array to be sorted is large.