Semester One of Academic Year (2014---2015) of BJUT

Data Structures and Algorithms I Exam Paper B

Exam Instructions: **Answer any 4 of 5 Questions. All questions carry equal marks.**

Honesty Pledge:

I have read and clearly understand the Examination Rules of Beijing University of Technology and am aware of the Punishment for Violating the Rules of Beijing University of Technology. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I would accept the punishment thereof.

Pledger: ___________ **Class No:** ___________

BJUT Student ID: ___________ **UCD Student ID**___________

∘∘

Notes:

The exam paper has 5 parts on 5 pages, with a full score of 100 points. You are required to use the given answer book only.

Part 1: Complexity and Sorting

1. What is the complexity of *Mergesort* in Big O notation?

(2 points)

2. Explain in your own words how the *Mergesort* algorithm works.

(6 points)

3. What is the complexity of *selection sort* in Big O notation?

(2 points)

4. Explain in your own words how the selection sort algorithm works. Illustrate with a small example (minimum 4 numbers).

(6 points)

5. Write the code for the *Mergesort* method. You do not need to write the merge method which has the signature **`merge(int[] f, int lb, int mid, int ub).`**

(9 points)

(Total 25 points)

Part 2: Singly Linked List

1. Name one advantage a *LinkedList* has over an array. Name one disadvantage.

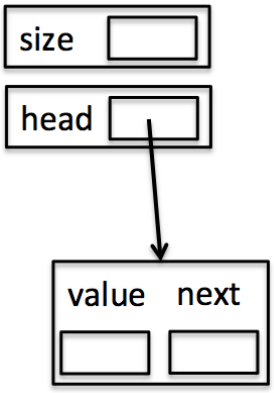
(4 points)

2. A singly linked list only stores the location of the head/top of the list. Describe in English the steps involved in finding the last element in the list. What is the complexity of this operation in big O notation?

(6 points)

3. Complete the diagram below to show the contents of the linked list after the following operations.

`addFirst("First"), addLast("Last"), n = first(), addAfter(n, "after"), l = last(), addAfter(l, "before"), n = next(n), addAfter(n, "next").`



(7 points)

4. Write the method *addBefore* for the `SinglyLinkedList` class. This method should take two parameters, the Node to be inserted before and the Object to be inserted.
   You can assume the top of the list is stored in the variable "head" and the number of elements in the variable "size".
   Remember to consider special cases

(8 points)
(Total 25 points)

Part 3: Queue

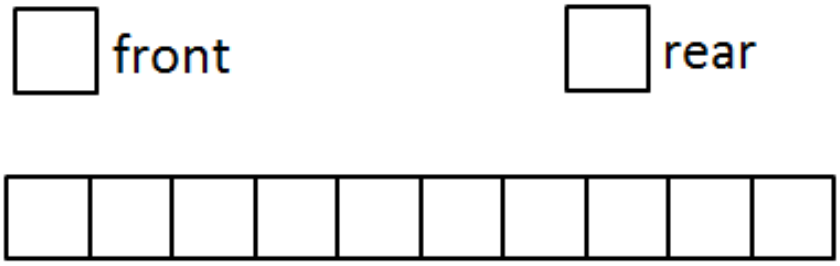
1. Describe the *Queue* abstract data type. What are the core operations?

(4 points)

2. Describe in English how the array-based implementation of the Queue works. What happens to the variables **`front`** and **`rear`** when items are added and removed?

(6 points)

3. Copy the diagram below and complete it to show the contents of the array based queue following the operations.
   `enqueue("A"), enqueue("C"), dequeue(), enqueue("Z"), enqueue("P"), dequeue(), enqueue("T"), enqueue("F")`



(7 points)

4. Write the method **`enqueue`** for the circular array implementation of the queue. This method should take one parameter: the Object to be inserted.
   You can assume the array is named "**head"**, and the top and bottom of the queue are stored in "**front"** and "**rear".**

(8 points)
(Total 25 points)

Part 4: Vector

1. What is a *Vector*? Name the core operations of a Vector.

(4 points)

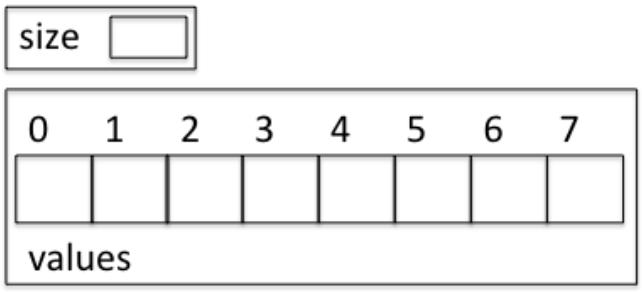
2. What is the complexity of the method **`insertAtRank`** for both array based implementation and link based implementation? Explain for each why this is.

(8 points)

3. Copy the picture below and complete it to show the result of the following operations.

```
insertAtRank(0,34), insertAtRank(1,12),
insertAtRank(1,14), replaceAtRank(0,18),
insertAtRank(0,155), insertAtRank(4,12),
removeAtRank(1)
```

(5 points)



4. Write the code for the method **`insertAtRank`**, this should be written for the array-based implementation of the Vector.

(8 points)

Part 5: Map

1. What is a *Map*? Name the core operations of a Map (you do not need to include the Iterators).

(5 points)

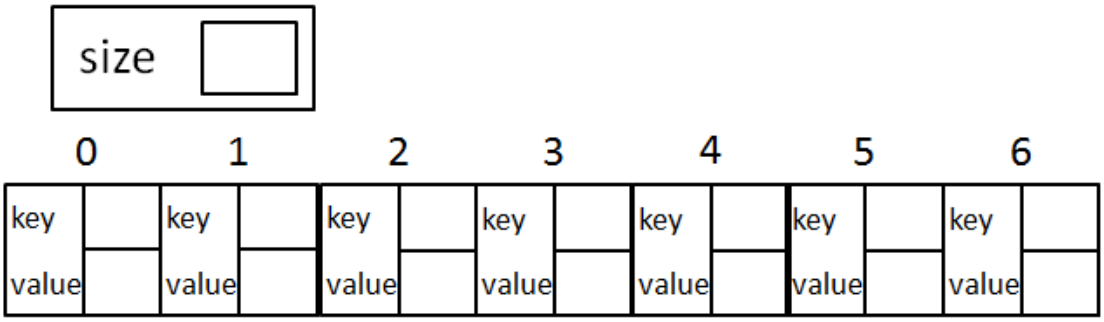
2. What are the two collision handling strategies you have studied. How do they both work?

(6 points)

3. Copy the picture below and complete it to show the result of the following operations when performed on a HashMap with the Linear Probing collision handling strategy. The array size in the HashMap is 7 and the **`hashCode`** method returns the key % 7.

```
put(12,34), put(21,19), put(12,67), put(11,15),
put(93,34), put(68,5), remove(93), put(40,56),
```

(6 points)



4. Write the code for the method **`put`**, this should be written for the hash map using *separate chaining* of the Map. The values are stored in an array of linked lists named **values.** You can use the method **`find`** which takes an array of linked lists and a key as parameters and returns the Position the entry is store in or null if it is not stored in the hashmap.

(8 points)

(Total 25 points)