

King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department

Second Semester 102 (2010/2011)

ICS 202 – Data Structures

Major Exam 1

Tuesday, 22nd March, 2011

Time: 90 minutes

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID# |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section 01 |  | Question # | Max Marks | Marks Obtained |
| Dr. Wasfi |  | 1 | 20 |  |
| 11-11:50am |  | 2 | 25 |  |
| Section 02 |  | 3 | 25 |  |
| Dr. Sami |  | 4 | 30 |  |
| 10-10:50am |  | Total | 100 |  |

**Instructions**

1. **Write your name and ID in the respective boxes above and circle your section.**
2. **This exam consists of 8 pages, including this page, plus one double-sided reference sheet, containing 4 questions.**
3. **You have to answer all 4 questions.**
4. **The exam is closed book and closed notes. No calculators or any helping aids are allowed.**
5. **Make sure you turn off your mobile phone and keep it in your pocket if you have one.**
6. **The questions are not equally weighed.**
7. **The maximum number of points for this exam is 100.**
8. **You have exactly 90 minutes to finish the exam.**
9. **Make sure your answers are readable.**
10. **If there is no space on the front of the page, feel free to use the back of the page. Make sure you indicate this in order not to miss grading it.**

Q.1: (20 points) Give short answers to each of the following:

1. (6 points) State the big-O complexity of each of these data structures for each of the following methods:

|  |  |  |
| --- | --- | --- |
|  | **extractLast** | **insertAfter** |
| **MyLinkedList** |  |  |
| **DoublyLinkedList** |  |  |
| **Arrays** |  |  |

1. (8 points) Consider the following postfix expression:

5 2 \* 12 4 / – 3 6 \* +

1. Show the unambiguous infix equivalent of the above expression.
2. Evaluate the postfix expression using a stack. Show the contents of the stack after each operation.
3. (6 points) Consider the following recursive method

**public int fib(int n){**

**if (n < 0) // this condition is checked every recursive call.**

**throw new IllegalArgumentException("Negative argument");**

**else if (n == 0 || n == 1)**

**return n;**

**else**

**return fib(n – 1) + fib(n – 2);**

**}**

Rewrite the recursive method above such that you do not have to check whether the parameter **n** is negative in each recursive call.

Q.2: (25 points) Let SearchableArray be a class that implements the SearchableContainer interface and features a static array of size MAX\_SIZE:

public class SearchableArray implements SearchableContainer {

public Object [] array;

int count;

public final static int MAX\_SIZE = 100;

public SearchableArray()

{

array = new Object[MAX\_SIZE];

count = 0;

}

...

}

1. (15 points) Implement method **insert** in SearchableArray knowing that the array should always be ordered in ascending order. Note that duplicates are allowed.

**public void insert(Comparable obj)**

**{**

**}**

1. (10 points) Implement method **withdraw** in Searchable array. Note that the SearchableArray should always be contiguous (it does not contain gaps).

**public void withdraw(Comparable obj)**

**{**

**}**

Q.3 (25 points) Consider the following piece of code:

sum = 0;

for (i=1; i<=n; i\*=2) {

for (j=1; j<=i; j++)

sum++; // MyStatement1

for (k=1; j<=n; k++)

sum++; // MyStatement2

}

1. (8 points) Count the number of times MyStatement1 gets executed in terms of n.
2. (8 points) Count the number of times MyStatement2 gets executed in terms of n.
3. (9 points) What is the time complexity of the above algorithm in terms of Big Oh notation in the worst case? Is it different from the best case? Justify your answer.

Q.4: (30 points) A priority queue is a queue in which the dequeue operation removes an item from the front of the queue but the enqueue operation inserts items according to their priorities. That is, a higher priority item is always enqueued before a lower priority element. An element that has the same priority as one or more elements in the queue is enqueued after all the elements with that priority.

Consider a priority queue implementation using a single linked list as follows:

**public class PriorityQueueAsLinkedList extends AbstractContainer implements Queue {**

**protected MyLinkedList list;**

**public PriorityQueueAsLinkedList(){ list = new MyLinkedList();**

**}**

**……**

**}**

Where MyLinkedList is defined as follows:

**public class MyLinkedList{**

**protected Element head;**

**protected Element tail;**

**public final class Element{**

**Object data;**

**int priority;**

**Element next;**

**Element(Object obj, int priorit, Element element){**

**data = obj;**

**priority = priorit;**

**next = element;**

**}**

**}**

**}**

Note that the inner class Element has an additional data member for the priority. For instance an Element with priority 1 is **higher** than an Element with priority 2.

As an example, this is the status of the PriorityQueueAsLinkedList after enqueuing Obj1, Obj2,Obj3, Obj4, and Obj5 with priorities 4,2,4,1, and 2 respectively.



1. (12 points) Assuming that **methods prepend, append, insertAfter**, and **insertBefore** have an additional parameter for the priority, provide the method enqueue of the class PriorityQueueAsLinkedList:

**public void enqueue(Object ob, int priorit)**

**{**

**}**

1. (3 points) What is the complexity of the enqueue method in the big O notation?
2. (12 points) Assume that the interpretation of the priority value has changed and that an Element with a smaller value will have lower priority than an Element with a higher value. For example, an Element with priority 3 should now be dequeued before an Element with priority 1. Write a method reverse in the PriorityQueueAsLinkedList class that updates the priority queue accordingly. For instance, the previous Queue after the call to reverse will be as follows:

****

**public void reverse()**

**{**

**}**

1. (3 points) What is the complexity of the reverse method in the big O notation?

**Quick Reference Sheet**

|  |  |
| --- | --- |
| public interface Iterator {  boolean hasNext( );  Object next( ) throws NoSuchElementException;  }  public interface Visitor {  void visit (Object object);  boolean isDone( );  }  public interface Container {  int getCount( );  boolean isEmpty( );  boolean isFull( );  void purge( );  void accept (Visitor visitor);  Iterator iterator( );  }  public interface SearchableContainer extends Container {  boolean isMember (Comparable object);  void insert (Comparable object);  void withdraw (Comparable obj);  Comparable find (Comparable object);  }  public class Association implements Comparable  public Association(Comparable key, Object val)  public Association(Comparable key)  public Comparable getKey( )  public Object getValue( )  public void setKey(Comparable key)  public void setValue(Object value )  public int compareTo(Object obj)  public boolean equals(Object obj)  public String toString( )  } | public class MyLinkedList {  public void purge( )  public Element getHead( )  public Element getTail( )  public Element find(Object obj)  public boolean isEmpty( )  public Object getFirst( )  public Object getLast( )  public void prepend(Object obj)  public void append(Object obj)  public void assign(MyLinkedList list)  public void extract(Object obj)  public void extractFirst( )  public void extractLast( )  public String toString( )  public Iterator iterator( )    public final class Element {  public Object getData( )  public Element getNext( )  public void insertAfter(Object obj)  public void insertBefore(Object obj)  public void extract( )  }  }  public interface Stack extends Container {  Object getTop( );  void push(Object obj);  Object pop( );  }  public interface Queue extends Container {  Object getHead( );  void enqueue(Object obj);  Object dequeue( );  } |

 ,  , , 

Q.4: (30 points) A priority queue is a queue in which the dequeue operation removes an item from the front of the queue but the enqueue operation inserts items according to their priorities. That is, a higher priority item is always enqueued before a lower priority element. An element that has the same priority as one or more elements in the queue is enqueued after all the elements with that priority.

Consider a priority queue implementation using a single linked list as follows:

**public class PriorityQueueAsLinkedList extends AbstractContainer implements Queue {**

**protected MyLinkedList list;**

**public PriorityQueueAsLinkedList(){ list = new MyLinkedList();**

**}**

**……**

**}**

Where MyLinkedList is defined as follows:

**public class MyLinkedList{**

**protected Element head;**

**protected Element tail;**

**public final class Element{**

**Object data;**

**int priority;**

**Element next;**

**Element(Object obj, int priorit, Element element){**

**data = obj;**

**priority = priorit;**

**next = element;**

**}**

**}**

**}**

Note that the inner class Element has an additional data member for the priority. For instance an Element with priority 1 is **higher** than an Element with priority 2.

As an example, this is the status of the PriorityQueueAsLinkedList after enqueuing Obj1, Obj2,Obj3, Obj4, and Obj5 with priorities 4,2,4,1, and 2 respectively.

