UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)

Academic Year 2008/2009 – 2nd Year Examination – Semester 4

IT4103: Programming II

PART 2 - Structured Question Paper

15th August, 2009

(ONE HOUR)

To be completed by the candidate

BIT Examination Index No: .....

Important Instructions:

- The duration of the paper is **1 (one) hour**.
- The medium of instruction and questions is English.
- This paper has **2 questions** and **6 pages**.
- **Answer all 2 questions. Questions do not carry equal marks.**
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.

Questions Answered

Indicate by a cross (×), (e.g. [X]) the numbers of the questions answered.

	Question Numbers		
To be completed by the candidate by marking a cross (×).	1	2	
To be completed by the examiners:			

Index No …….

1) **Write descriptions on the following and introduce required segments of code when necessary.**

a) Assume that there is a database, having the name **Sales,** which is created by using Microsoft Access, having one table in it, with the name **Customer**. The Customer table has three fields namely, Customer Id, Customer Name and Address. Write a Java program to show the database connectivity using the Sales database. It is not required to consider the creation of an Interface to show the data.

(30 Marks)

ANSWER IN THIS BOX

```
import java.sql.Connection;
import java.sql.DriverManager
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
public class MakingAStatement {
public static void main(String[] args) {
// Load the driver
try {
// Load the driver class
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// This defines the data source for the driver
String sourceURL = new String("jdbc:odbc:Sales");
// Create connection through the DriverManager
Connection databaseConnection =
DriverManager.getConnection(sourceURL);
Statement statement = databaseConnection.createStatement();
ResultSet authorNames = statement.executeQuery(
"SELECT lastname, firstname FROM authors");
} catch(ClassNotFoundException cnfe) {
System.err.println(cnfe);
} catch(SQLException sqle) {
System.err.println(sqle);
   }
 }
}
```

b) Write an introductory paragraph on the sorting method called Quick Sort. Use the necessary segment of codes in Java to show the implementation of Quick Sort. **(30 marks)**

ANSWER IN THIS BOX

```
class ArrayIns
{
private double[] theArray;
private int nElems;
public ArrayIns(int max) {
theArray = new double[max];
nElems = 0;
}
public void insert(double value) {
theArray[nElems] = value; // insert it
nElems++; // increment size
}
public void display() // displays array contents
{
System.out.print("A=");
for(int j=0; j<nElems; j++) // for each element,
System.out.print(theArray[j] + " "); // display it
System.out.println("");
}
public void quickSort()
{
recQuickSort(0, nElems-1);
}

public void recQuickSort(int left, int right)
{
if(right-left <= 0) // if size <= 1,
return; // already sorted
else // size is 2 or larger
{
double pivot = theArray[right]; // rightmost item
// partition range
int partition = partitionIt(left, right, pivot);
recQuickSort(left, partition-1); // sort left side
recQuickSort(partition+1, right); // sort right side
})

public int partitionIt(int left, int right, double pivot)
{
int leftPtr = left-1; // left (after ++)
int rightPtr = right; // right-1 (after --)
while(true)
{ // find bigger item
while(theArray[++leftPtr] < pivot)
; // (nop)
// find smaller item
while(rightPtr > 0 && theArray[--rightPtr] > pivot)
; // (nop)
if(leftPtr >= rightPtr) // if pointers cross,
break; // partition done
else // not crossed, so
swap(leftPtr, rightPtr); // swap elements
} // end while(true)
swap(leftPtr, right); // restore pivot
return leftPtr; // return pivot location
} // end partitionIt()
```

```
public void swap(int dex1, int dex2) // swap two elements
{
double temp = theArray[dex1]; // A into temp
theArray[dex1] = theArray[dex2]; // B into A
theArray[dex2] = temp; // temp into B
} // end swap(
}
```

Quicksort is very popular sorting algorithm, and for good reason: in the majority of situations, it's the fastest, operating in O(N*logN) time. (This is only true for *internal* or in-memory sorting; for sorting data in disk files other methods may be better.)
Quicksort was discovered by C.A.R. Hoare in 1962.

Basically the quicksort algorithm operates by partitioning an array into two subarrays, and then calling itself to quicksort each of these subarrays. However, there are some embellishments we can make to this basic scheme. These have to do with the selection of the pivot and the sorting of small partitions.

To understand quicksort, its better to familiar with the partitioning algorithm also.

2) Consider the following scenario:

The word 'queue' is used in Britain to denote a line. A line of persons at the Cinema is an example to show a queue in real life. The first person to join the line is the first to buy a ticket. The last person to join the line is the last to buy a ticket.

One can see the following behaviour in a line at a Cinema.

- Spectators joining the rear end of the line.
- After taking a ticket leaving the line.
- Whenever required, the ticket issuer calling the spectator who is at the front end of the line.
- A situation where there are no spectators in the line.
- A situation where there are spectators in the line.

After considering the above scenario, one has to identify a suitable data structure which can be used to implement the above mentioned behaviour as functions in a computer environment.

Do the following:

a) Write the name of the data structure which can be used in a computer environment to manipulate data having similar behaviour as the above scenario. **(02 Marks)**

ANSWER IN THIS BOX

Queue data structure

b) Implement the identified data structure using Java programming language following Object Oriented concepts. **(38 Marks)**

ANSWER IN THIS BOX

```
import java.io.*;
class Queue
{
private int maxSize;
private int[] queArray;
private int front;
private int rear;
private int nItems;
public Queue(int s) // constructor
{
maxSize = s;
queArray = new int[maxSize];
front = 0;
rear = -1;
nItems = 0;
}
public void insert(int j) // put item at rear of queue
{
if(rear == maxSize-1) // deal with wraparound
rear = -1;
queArray[++rear] = j; // increment rear and
insert
nItems++; // one more item
}
public int remove() // take item from front of queue
{
int temp = queArray[front++]; // get value and incr front
if(front == maxSize) // deal with wraparound
front = 0;
nItems--; // one less item
return temp;
}
public int peekFront() // peek at front of queue
{
return queArray[front];
}
public boolean isEmpty() // true if queue is empty
{
return (nItems==0);
}
public boolean isFull() // true if queue is full
{
return (nItems==maxSize);
}
public int size() // number of items in queue
{
return nItems;
}
} // end class Queue
```

****