UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY

Academic Year 2010/2011 –2nd Year Examination – Semester 4

IT4104: Programming II

Part 1: Multiple Choice Question Paper

6th August 2011
(ONE HOUR)

Important Instructions :

- The duration of the paper is **1 (one) hour.**
- The medium of instruction and questions is English.
- The paper has **25 questions** and **7 pages**.
- All questions are of the MCQ (Multiple Choice Questions) type.
- All questions should be answered.
- Each question will have 5 (five) choices with **one or more** correct answers.
- All questions will carry equal marks.
- There will be a penalty for incorrect responses to discourage guessing.
- The mark given for a question will vary from 0 (*All the incorrect choices are marked & no correct choices are marked*) to +1 (*All the correct choices are marked & no incorrect choices are marked*).
- Answers should be marked on the special answer sheet provided.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.
- Mark the correct choices on the question paper first and then transfer them to the given answer sheet which will be machine marked. **Please completely read and follow the instructions given on the other side of the answer sheet before you shade your correct choices.**

1) Select from among the following, the statement(s) which is/are **not** true on skip lists.

(a) Skip lists perform sequential scanning to locate searched-for elements. (b) Skip lists facilitate data to be ordered but still sequential search is required for locating elements. (c) Skip lists allow avoiding certain nodes in search. (d) One can not always expect a reference field in nodes in a skip list. (e) By using a skip list one can create a spanning tree.

2) Select among the following, different approaches one can expect to organize self organizing lists.

(a) Move-to-front	(b) Transpose	(c) Count
(d) Sequential search	(e) Ordering	

3) Consider the following segment of code written in Java.

```
Public Object deleteHead{
       Object el = head.infor;
       Head = head.next;
       Return el;
  }
```

Then consider the following table listing some basic data related operations.

Identifier	Data related Operations
A	Add
B	Update
C	Delete
D	Display

In the following table names of 4 data structures are listed.

Identifier	Data Structure
1	Array
2	Singly Linked List
3	Graph
4	Tree

Select from among the following, the correct operation/s and correct data structure/s which is/are referred in the given code segment.

(a) A → 1	(b) D → 4	(c) C → 2
(d) B → 3	(e) C → 3	

4) Consider the following description on a kind of a linked list.

"the list forms a ring and that list is finite and each node has a successor"

Select from among the following, the correct list type which is mentioned in the description.

(a) Singly Linked List	(b) Doubly Linked List	(c) Skip List
(d) Circular List	(e) Self Organizing List	

5) Consider the following pseudocode.

```
whatSort(digraph)
   for i = 1 to [v]
   find a minimal vertex v;
   num(v) = i;
   remove from digraph vertex v and all edges incident with v;
```

Select from among the following, the task which can be achieved by the pseudocode.

(a) Insertion sort	(b) Selection sort	(c) Quick sort
(d) Topological sort	(e) Shell sort	

6) Select from among the following, the correct algorithm which can be used for connectivity in Directed Graphs.

(a) Depth First	(b) Kruskal	(c) Dijkstra
(d) Tarjan	(e) Greedy	

7) Consider the following real life scenarios.

A → Waiting in a line to buy a ticket to see a movie
B → Placing Compact Disks in a CD folder
C → Filling gas into gas cylinders
D → Waiting to see a doctor to get treatment at a private medical centre
E → Sequencing aeroplanes in a runaway

Select from among the following, the correct access mechanisms (whether FIFO – Frist in First Out or LIFO – Last in First Out) that can be used with the above scenarios.

(a) A → LIFO	(b) B → FIFO	(c) C → LIFO
(d) D → FIFO	(e) E → LIFO	

8) Select from among the following, examples which can be considered as Stack related behaviours.

(a) Behaviour of memory in an execution of a recursion
(b) Behaviour of a printer buffer
(c) Converting a decimal number to its binary representation
(d) Connecting a printer to a computer
(e) Converting infix expressions to postfix expressions

9) Select from among the following real life situations, where sorting can be applied.

(a) Informing the students of their grades in a University
(b) Cutting matured trees from a forest
(c) Calling employee who are coming from Piliyandala
(d) Searching for a book in a Library
(e) Looking for a patient in a hospital

10) Select from among the following, methods which can be used to solve the shortest path in a Graph.

(a) Label setting	(b) Method calling	(c) In order
(d) Depth first traversal	(e) Label correcting	

11) Consider the following pseudo code.

```
problem(v)
  num(v) = i++;
    for all vertices u adjacent to v
           if num(u) is 0
                 pred(u) = v;
                   problem(u);
          else if num(u) is not ∞
                pred(u) = v;
                    problem solved;
       num(v) = ∞;
```

Select from among the following, the suitable application/s that can be expected from the above pseudo code.

(a) Detecting cycles in a Graph
(b) Creating an adjacency list of a Graph
(c) Finding the shortest path of a Graph
(d) Inspecting a spanning tree in a Graph
(e) Performing a Depth First search of a Tree

12) Select from among the following, the correct statements on spanning trees in Graphs.

(a) As a result of cycles in a Graph one can expect alternative paths in it.
(b) Creation of a minimum number of connections in a Graph is called creation of a spanning tree.
(c) A popular algorithm devised to create a spanning tree is called Radix.
(d) The Kruskal's algorithm requires that all the edges be ordered before creating the spanning tree.
(e) The algorithm devised by Dijkstra dose not requires ordered edges to make a spanning tree.

13) Select from among the following, the information that can be expected in an activation record in recursion.

(a) Value for all parameters to the method, location of the first cell if an array is passed or a variable is passed by reference and copies of all other data items
(b) Local variables that can be stored elsewhere, in which case, the activation record contains only their descriptors and pointers and pointer to the location where they are stored.
(c) The return address to resume control by the caller, the address of the caller's instruction immediately following the call
(d) A dynamic link, which is pointer to the caller's activation record
(e) The returned value for a method is not declared as void. Because the size of the activation record may vary from one call to another, the returned value is placed right above the activation record of the caller.

14) Consider the following segment of code written to check whether the particular data structure is full.

```
public boolean isFull(){
  return first == 0 && last == size - 1 || first == last - 1;
   }
```

Select from among the following, the data structure which was in focus when writing the given code.

(a) Queue (b) B-tree (c) Binary tree
(d) Stack (e) Linked List

Consider the following set of operations executed on a particular data structure to answer questions 15 and 16.

clear()
enqueue(5)
enqueue(7)
enqueue(6)
clear()
enqueue(1)
enqueue(3)
firstEl()
dequeue()
enqueue(9)
enqueue(3)
firstEl()
dequeue()
enqueue(4)

15) Select from among the following, the data structure which possesses the operations shown in the above list.

(a) Stack	(b) Queue	(c) Array
(d) Linked List	(e) Graph	

16) After executing the statements successfully, what would the values inside the data structure be?

(a) 567	(b) 12	(c) 93
(d) 4	(e) 934	

17) Select from among the following, correct statement/s on Trees.

(a) Each node of a Tree has to be reachable from the root.
(b) Sequence of arcs which follow to reach a leave from the root is called path.
(c) The level of a node is determined by the length of the path from the root to the node.
(d) The height of a non empty Tree is the maximum level of a node in the Tree.
(e) The empty Tree is a legitimate Tree of height 0(by definition).

18) Select from among the following, valid option/s which contribute to the efficiency of B-trees in a positive manner.

(a) Having more records per node	(b) So many nodes per level
(c) All the nodes at least half full	(d) Less height
(e) Fewer accesses per data insertion.	

19) Select from among the following, valid statement/s on different data structures.

(a) Linked Lists usually provide greater flexibility in manipulating data than arrays.
(b) Linked Lists can be used to organize objects representing them hierarchically.
(c) Stacks and Queues are limited to only one dimension in representing objects.
(d) An array can be considered as a linear data structure.
(e) An array cannot be used to represent Objects in a hierarchical manner.

20) Consider the following pseudocode taking note of the blank indicated as **BLANK.**

```
selectionSort(data[])
     for i = 0 to data.length – 2
               BLANK
          swap it with data[i];
```

Select from among the following, the valid statement to fill the blank.

(a) Comparable tmp = (Comparable)data[i];
(b) swap the root with the element in position i;
(c) select the smallest among data[i]…….data[data.length – 1]
(d) include element in either in subarray1 = {el: el <= bound} or subarray2 = {el: el <= bound};
(e) swap elements in position j and j – 1 if they are not in order

21) Consider the following pseudocode taking note of the blank indicated as **BLANK**.

```
public void insertionSort(Object[] data){
   for(int i = 1,j; i < data.length; i++){
                    BLANK
      for(j = i ; j > 0 && tmp.compareTo(data[j-1]) < 0;j--)
          data[j] = data[j-1];
       data[j] = tmp;
 }
}
```

Select from among the following, the valid statement to fill the blank.

(a) Comparable tmp = (Comparable)data[i];
(b) swap the root with the element in position i;
(c) select the smallest among data[i]…….data[data.length – 1]
(d) include element in either in subarray1 = {el: el <= bound} or subarray2 = {el: el <= bound};
(e) swap elements in position j and j – 1 if they are not in order

22) Consider the following pseudocode taking note of the blank indicated as **BLANK**.

```
bubbleSort(data[])
    for i = 0 to data.length – 2
      for j = data.length – 1 downto  i + 1
                BLANK
```

Select from among the following, the valid statement to fill the blank.

(a) Comparable tmp = (Comparable)data[i];
(b) swap the root with the element in position i;
(c) select the smallest among data[i]…….data[data.length – 1]
(d) include element in either in subarray1 = {el: el <= bound} or subarray2 = {el: el <= bound};
(e) swap elements in position j and j – 1 if they are not in order

23) Consider the following pseudocode taking note of the blank indicated as **BLANK**.

```
heapSort(data[])
    transform data into a heap;
      for i = data.length – 1 downto 2
                          BLANK
          restore the heap property for the tree data[0]……data[i-1];
```

Select from among the following, the valid statement to fill the blank.

(a) Comparable tmp = (Comparable)data[i];
(b) swap the root with the element in position i;
(c) select the smallest among data[i]…….data[data.length – 1]
(d) include element in either in subarray1 = {el: el <= bound} or subarray2 = {el: el <= bound};
(e) swap elements in position j and j – 1 if they are not in order

24) Consider the following pseudocode taking note of the blank indicated as **BLANK**.

```
quickSort(array[])
   if array.length > 1
     choose bounds;
         while there are elements left in array
                 BLANK
      quickSort(subarray1);
      quickSort(subarray2);
```

Select from among the following, the valid statement to fill the blank.

(a) Comparable tmp = (Comparable)data[i];
(b) swap the root with the element in position i;
(c) select the smallest among data[i]…….data[data.length – 1]
(d) include element in either in subarray1 = {el: el <= bound} or subarray2 = {el: el <= bound};
(e) swap elements in position j and j – 1 if they are not in order

25) Select from among the following, a good explanation for Hash functions.

(a) By using hash functions one can create unique indexes to store unique data items.
(b) By using hash function one can store data having the same type.
(c) Hash functions can be accessed in First in First out manner.
(d) By using a hash function the spanning tree can be calculated.
(e) Hash functions make the circular queues more efficient by means of accessing its data.

****