The Open University of Sri Lanka
B.Sc Degree Programme – Level 04
Final Examination – 2008/2009
CSU 2279 – Data Structures and Algorithms – Paper II
Duration: Two and Half hours
Date: 20th July 2009 Time: 10.00 a.m – 12.30 p.m

Answer **Four** Questions **Only**.

Q1. (i) Construct a binary tree where the *post-order* and *in-order* traversals are as follows;

Post-order	: D B I J G H E F C A
In-order	: D B A I G J E H C F

(ii) Give the pre-order and level-order traversals of the above binary tree
(***Hint:*** level-order traversal visits all the nodes at the same level at a time, starting with level 0)

(iii) Write a Pascal procedure to insert a right child to the node 'A' of the above binary tree that you constructed in part (i).

(iv) Represent the following mathematical expressions using binary trees.

(a) a + b! * c

(b) (p + q * r) $ ((p! + s) * r) : Here, the $ sign represents the exponentiation

(c) (x + log y) / (p + r) * (a + b)

Q2. (i) Briefly explain the *Sequential Bitstring Representation of Sets* with examples.

(ii) Give the definition of a character set using the above method of representation. (You may use the array based implementation of a set.)

(iii) Write appropriate Pascal functions/procedures to simulate the following set operations. State clearly the assumptions if any.

(a) A procedure to insert an element into the set S1.

(b) A procedure to delete an element from the set S1.

(c) A function **IsIdentic(S1, S2)** that returns true if the set S2 is identical to the set S1.

(d) A procedure **Intersect(S1,S2)** which changes the set S1 into the intersection of sets S1 and S2.

(e) A function **IsNull(S)** which returns true if the set S is a null set otherwise returns false.

(f) A function **MutualEx(S1, S2)** returns true if the two sets S1 and S2 are mutually exclusive, otherwise returns false.
(**Hint** :Two sets whose intersection is empty are said to be Mutually exclusive.
You may use the above procedures **Intersect (S1, S2),** and **IsNull(S).)**

Q3. (i) Using an appropriate diagram, describe the differences between the array implementation of a *stack* and pointer implementation of a *stack*.

(ii) Indicate whether a *stack* would be a suitable data structure for each of the following applications. Justify your answers.

(a) A program to receive data that are to be saved and processed in the reverse order.

(b) A word processor to have a special key that causes the preceding command to be displayed.

(c) A programme to keep track of patients as they check–in into a medical clinic, assigning patients to doctors on a first-come first-served basis.

(d) A data structure used to keep track of the return addresses for nested functions while a programme is running.

(iii) Use the definition of the following ADT to create functions/procedures to simulate stack operations from (a) to (c) given below.

```
type
       stackitem = integer;
       stack = record
            top : 0..maxlength ;
            data: array [1..maxlength] of stackitem;
       end;
```

(a) If the stack is full, then return true, or else return false.

(b) To delete an item from the stack.

(c) To insert an element into the stack.

Q4. (i) Assume that there is no simple data type in your Pascal version to represent a character string. Describe a suitable data structure to implement character strings in this Pascal version.

(ii) Using the above method write appropriate functions/procedures to simulate the following string operations.

(a) **Occur(S1,LETTER),** a function which returns the number of occurrences of the given letter, LETTER in the string S1.

(b) **Reverse(S1, S2),** a procedure which writes the reverse order of the string S1 in S2.

(c) **IsIdentic(S1, S2),** a function which returns true if the two strings S1 and S2 are identical, otherwise returns false.

(d) **IsPalin(S),** a function to check whether the given string S is a Palindrome. **(Hint** : A Palindrome is a word which reads the same backward as forward. Eg. MADAM .)

Q5. (i) Explain the differences between the pointer based and array based implementation of a *List* data structure.

(ii) Explain the process of,

(a) deleting an element from the linked list

(b) inserting an element into the linked list

by means of appropriate diagrams. (Show the pointer manipulation clearly)

(iii) Use the following Pascal declaration of a *singly-linked list* to answer the questions (iii) (a) to (d).

```
type
     celltype = record
          element   : integer ;
          next          : ^ celltype
     end;

     LIST = ^ celltype ;
Var
     L : LIST
```

(a) Write a procedure **Insert(x, p, L)** which places an element 'x' at the position 'p' into the list' L'.

(b) Write a function **Next(p, L)** which returns to the following position of 'p' on list L.

(c) Write a function **Locate(x, L)** which locates and returns the position of the element 'x' in the list 'L'.

(d) Change and rewrite the given declaration of the *linked list* so that it implements a *doubly-linked list.*

Q6. (i) Explain the concept of 'Circular array implementation of a Queue data structure'

(ii) Assume that a circular array of a queue has a *maxlength* places. Use appropriate diagrams to explain why it is restricted the queue grow not longer than *maxlength-1.*

(iii) Give the Pascal declaration to implement the above *queue* data structure.

(iv) Using the above (Part (iii)) declaration, write appropriate functions/procedures to simulate the following queue operations. State clearly the assumptions if any.

(a) A function **FRONT (Q)**: which returns the first element of the queue 'Q'.

(b) A procedure **ENQUEUE (Q, x)**: which inserts an element 'x' into the queue 'Q'.

(c) A Procedure **DEQUEUE (Q)**: which deletes an element from the queue 'Q'.

(d) A procedure **CONTENT (Q)**: which displays the content of the queue 'Q'.