# _LITfinalLOGO

# **SUMMER EXAMINATIONS 2016**

**Tuesday, 10th May 2016, 14.30 p.m. – 16.30 p.m.**

**KSDEM\_8\_Y2**

**Course:** Bachelor of Science (Hons) in Software Development

**Year:** Two

**Subject:** Data Structures & Algorithms

**Time Allowed:** 2 Hours

**Instructions:**

**1.** Answer **any THREE (3)** Questions.

1. All questions carry equal marks
2. The examination paper is marked out of 75 marks.
3. Start each question on a new page.
4. Write the question number at the top of each page.
5. Circle the numbers of the questions you answer at the

front of your answer book.

**Additional Attachments or Exam Material to accompany this paper:**

**A.** None

**Internal Examiners: External Examiners:**

Mr. Des O’Carroll Ms Una L’Estrange

**Q.1.**

**(a)** Describe, with the aid of a diagram, how an item is deleted from an unsorted linked list. Your answer should list the different cases and explain what action to take for each case.

**(10 marks)**

**(b)** Given the following class declaration for an Unsorted Linked List:

template <class ItemType>

struct NodeType;

template <class ItemType>

class UnsortedType

{

public:

UnsortedType();

~UnsortedType();

bool IsFull() const;

int LengthIs() const;

void MakeEmpty();

void RetrieveItem(ItemType& item, bool& found);

void InsertItem(ItemType item);

void DeleteItem(ItemType item);

void ResetList();

void GetNextItem(ItemType& item);

private:

NodeType<ItemType>\* listData;

int length;

NodeType<ItemType>\* currentPos;

};

The structure NodeType is as follows:

**template<class ItemType>**

**struct NodeType**

**{**

**ItemType info;**

**NodeType<ItemType>\* next;**

**};**

Implement the following member functions:

(i) void MakeEmpty( );

// Function: Initializes list to empty state.

// Post: List is empty.

**(4 marks)**

(ii) void RetrieveItem(ItemType& item, bool& found);

// Function: Retrieves list element whose key matches item's key (if

// present).

// Pre: Key member of item is initialized.

// Post: If there is an element someItem whose key matches item's // key, then found = true and item is a copy of someItem; otherwise

// found = false and item is unchanged.

// List is unchanged.

**(11 marks)**

**(Total 25 Marks)**

**Q.2.**

**(a)** Explain what is meant by

(i) tail recursion

(ii) procedure activation record (also known as **stack frame**)

**(2 X 2 marks)**

**(b)** Write recursive functions that will:

1. Calculate n! (factorial n) for some positive integer n.

**(3 marks)**

1. Reverse the digits in a positive integer number (called num)

-The header function must be: int reverse(int num)

**(5 marks)**

**(c)** Implement a non-tail-recursive version of the Towers of Hanoi function.

**(6 marks)**

**(d)** Trace through the Towers of Hanoi problem for a problem of size 3, by

working through the non-tail-recursive code. Your solution must show the values in the variables at all stages and clearly show the recursive calls to the Towers of Hanoi function. In addition the changes to the disks on the poles must be clearly identified as the trace progresses.

**(7 marks)**

**(Total 25 Marks)**

**Q.3.**

**(a)** Give the recursive definitions for the following:

1. a binary tree,

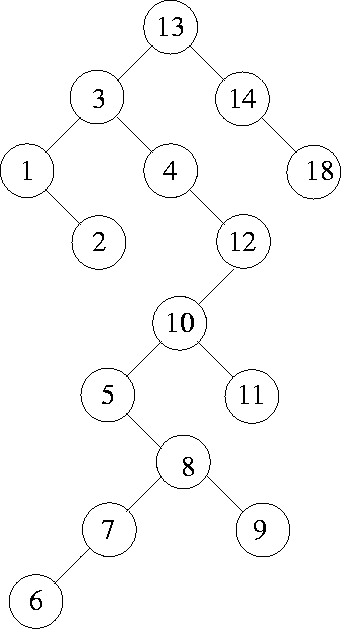
**(2 marks)**

1. a binary search tree.

**(4 marks)**

**(b)** Determine the order in which the nodes will be visited in:

* **PREORDER** traversal.
* **POSTORDER** traversal

****

**(2 X 3 marks)**

**(c)** Assuming the following class declaration for a Binary Search Tree ADT:

template <class ItemType>

struct TreeNode

{

ItemType info;

TreeNode \*left;

TreeNode \*right;

};

// Assume “<” and “==” operators have been overloaded for

// the class ItemType

template <class ItemType>

class TreeType

{

public:

TreeType( );

~TreeType( );

TreeType (const TreeType<ItemType>& originalTree);

void operator=(const TreeType<ItemType>& originalTree);

void MakeEmpty( );

bool IsEmpty( ) const;

bool IsFull( ) const;

int NumberOfNodes( ) const;

void RetrieveItem(ItemType &item, bool &found);

void InsertItem(ItemType item);

void DeleteItem(ItemType item);

void PrintTree(ostream &outFile) const;

private:

TreeNode<ItemType> \*root;

};

Implement the following recursive member functions, including any helper functions required:

1. int NumberOfNodes() const;

// Returns the number of nodes in the tree

**(6 marks)**

(ii) void InsertItem(ItemType item);

*//Function*: Adds item to tree.

*//Precondition*: Tree is not full. item is not in //tree.

*//Postcondition*: item is in tree. Binary search //property is maintained.

**(7 marks)**

**(Total 25 Marks)**

**Q.4.**

**(a)** Describe how an array can be used to implement a heap that is in the form of a complete binary tree. State the relationship between parent and child nodes. Support your explanations with a suitable example.

**(7 marks)**

**(b)** Describe in detail how the priority of an element on a priority queue might be changed.Support your explanations with a suitable example.

**(7 marks)**

**(c)** Implement the function ChangePriority( ), to change the priority of an element (oldItem) on a priority queue, by replacing it with (newItem) . Use the following function header:

template<class ItemType>

void PQType<ItemType>::ChangePriority(ItemType& oldItem, int index, ItemType newItem)

The function must take appropriate measures to restore the shape and/or order properties as appropriate.

**(11 marks)**

**(Total 25 Marks)**