Reg. No.

MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

A Constituent Institute of Manipal University

THIRD SEMESTER B.TECH. (INFORMATION TECHNOLOGY/COMPUTER AND COMMUNICATION ENGINEERING)

END SEMESTER EXAMINATIONS, NOVEMBER 2017

SUBJECT: DATA STRUCTURES [ICT-2103]

(REVISED CREDIT SYSTEM)
(21/11/2017)

TIME: 3 HOURS **MAX. MARKS: 50**

Instructions to candidates:
- Answer **ALL** questions.
- Missing data, if any, may be suitably assumed.

1A. Write a C++ menu driven program to do the following:
i. Create a singly linked list to store student details (Registration number and marks)
ii. Display the created student list
iii. Sort the student list in descending order based on the marks using the best sorting technique which can be used while working with linked list. **(05)**

1B. Write a user defined function to convert an infix expression to prefix form. Convert the following infix expression into prefix using stack and show the contents of stack at each step.
Infix expression: (a + b) * (c * d - e) * f / g **(03)**

1C. Construct a binary tree given its inorder and preorder sequence as 15, 30, 35, 40, 45, 50, 60, 70, 72, 75, 77, 86 and 50, 30, 15, 40, 35, 45, 70, 60, 86, 75, 72, 77 respectively. Also write the postorder traversal sequence for the constructed binary tree. **(02)**

2A. Write a C++ program to do the following:
i. Define a class Sparse which has row, column and value as data members, relevant member functions to initialize the data members and display methods.
ii. Initialize method should accept a 2D matrix from the user and check whether the matrix is sparse or not. If sparse then create object instance of a Sparse class.
iii. Sparse class should also include a user defined method to obtain fast transpose of a sparse matrix. **(05)**

2B. What is an expression tree? Write a function to construct an expression tree from a given postfix expression. **(03)**

2C. List the types of data structure with a real time application for each listed data structure type. **(02)**

3A. Write a C++ program with the following user defined functions:
i. Create doubly linked list to store polynomials.
ii. Add two polynomials and store the result as another doubly linked list.
iii. Display the contents of doubly linked list. **(05)**

3B. Write a function to check the depth of a binary tree and then print the ancestors of the tree in case the depth of the tree is greater than 2. **(03)**

3C. Write the answers for the following questions with proper justification.

i. What will be the content of Q after the execution of function if the input Q contents are 2, 4, 6, 8, 10.

```
void fun(Queue *Q)
{
 Stack S;
 while(!isEmpty(Q))
     push(&S, deQueue(Q));
 while(!isEmpty(&S))
     enQueue(Q, pop(&S));
}
```

ii. What is the output of the following program? What does the following fun() do in general?

```
int fun(int a[],int n)
{
 int x;
 if(n == 1)
    return a[0];
 else
    x=fun(a, n-1);
 if(x>a[n-1])
    return x;
 else
    return a[n-1];
}
void main()
{
 int arr[] = {12, 10, 30, 50, 100};
 cout<<fun(arr, 5);
}
```

(02)

4A. Write a complete class definition to implement multiple stacks in a single one dimensional array where the number of stacks and size of each stack is given as input by the user. Write the member functions for pushing element into ith stack, popping element from ith stack and displaying elements in the ith stack. **(05)**

4B. Write a user defined function to print the inorder traversal sequence for the threaded binary tree representation of the binary tree given in Figure Q.4B. Also, convert the binary tree into a threaded binary tree. **(03)**

4C. Define time complexity of a program. Determine the time complexity of a function to multiply two matrices using tabular method. **(02)**

5A. Write a user defined function to perform heap sort and show the steps involved in sorting the array 67, 34, 90, 2, 13, 45, 123 using heap sort. **(05)**

5B. Given a Binary Search Tree(BST) and two numbers K1 and K2, write a user defined function for printing all the elements of Binary Search Tree in the range K1 and K2. **(03)**

5C. Draw the adjacency matrix for the graph given in Figure Q.5C. Also write the Depth First Search (DFS) sequence starting from vertex 0. **(02)**

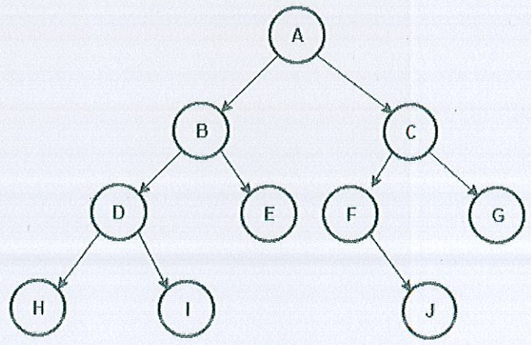

Figure: Q.4B

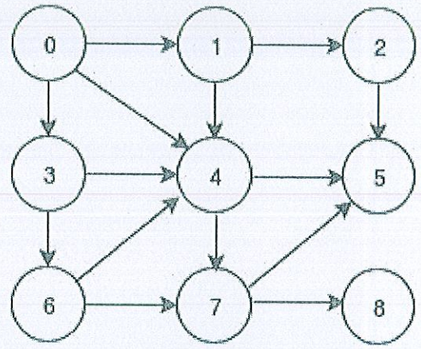

Figure: Q.5C