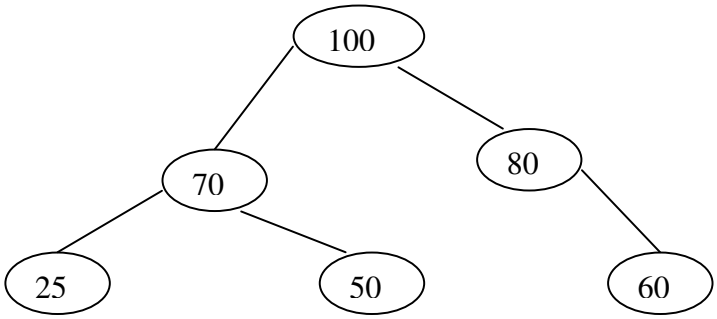
CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test - II

Sub:	DATA STRUCTURES & APPLICATIONS							Code:	15CS33
Date:	03 / 11 / 2016	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	ISE

Answer Any **FIVE** Complete Questions. Support answers with appropriate diagrams.

		Marks	OBE CO	OBE RBT
1 (a)	How do we create a node for a doubly linked list in C?	[03]	CO2	L3
(b)	Write a C function to delete a node whose info field is given from a doubly linked list.	[07]	CO2	L3
2 (a)	Write a C function to insert an element into a Circular Queue using Dynamic Arrays	[04]	CO2	L3
(b)	Write recursive C functions for the following tree traversals i) Inorder ii) Preorder iii) Postorder	[06]	CO2	L3
3 (a)	With reference to the tree given below answer the following questions: (i) Is it a binary tree? (ii) Is it a complete tree? (iii) Where will the left child of the node with key 60 point to, if it is converted into a Two-way In-threaded binary tree. 	[03]	CO4	L3
(b)	Suppose the following list of letters is inserted one after the other into an empty binary search tree: **J, R, D, G, T, E, M, H, P, A, F, Q** : i) Find the final tree **T** ii) Find the tree **T** after the node **M** is deleted iii) Find the tree **T** after the node **D** is also deleted.	[03+02+02]	CO4	L3
4	Write C functions to insert a node at the **front** and the **rear** end in a **circular linked list**. Show the sequence of steps with diagrams.	[05+05]	CO2	L3

5 (a)	Write a C function to insert a node in a Binary Search Tree. [05]	CO2	L3
(b)	Give the algorithm for Breadth First Search(BFS) in a graph. [05] Trace the algorithm and demonstrate how BFS works for any graph of your choice.	CO5	L3
6 (a)	Given the following empty Circular QUEUE: __ , __ , __ , __, __ [10] with FRONT=NULL, REAR=NULL. Perform the following in sequence. i) Insert **A,B,C** ii) Delete **A** iii) Insert **D, E** iv)Insert **F** v) Delete **B,C** vi) Insert **G,H** vii) Delete **E,F** viii) Insert **K** ix) Delete **G,H** x) Delete **K**	CO5	L3
7 (a)	A binary tree T has **9** nodes. The inorder and preorder traversals of T yield the following sequence of nodes: [04] **Inorder: EACKFHDBG** **Preorder: FAEKCDHGB** Draw the tree **T**	CO6	L3
(b)	Draw the inorder threading of the above binary tree T. Define the structure of a node of this tree in C. [03+03]	CO1	L3

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Acquire the knowledge of data type, data structure, and operations on data structure and file structure.	1	1	2	2	2	2	0	0	0	0	0	0
CO2:	Design, implement and analyze the concepts of data structures using C-Programming language.	1	1	1	2	1	1	0	0	0	0	0	0
CO3:	Implement linear data structures such as stacks, queues and linked lists to solve various computing problems.	1	1	1	1	2	1	2	0	0	0	0	2
CO4:	Implement non-linear data structures such as trees and graphs to solve various computing problems.	1	1	1	1	2	1	2	0	0	0	0	2
CO5:	Identify and apply the appropriate data structure that efficiently models the information in a problem.	1	1	1	1	2	1	2	0	0	0	0	2
CO6:	Design and implement new algorithms to solve real world problems.	1	1	1	1	2	1	1	2	2	2	2	2

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*

Q.1 (a). /* Doubly Linked List Node */

```
struct node
{
    int info;
    struct node *llink;
    struct node *rlink;
};
typedef struct node *NODE;
```

NODE first; or
struct node *first;

(3)

Q.1 (b) /* C func. to delete a node whose info field is given */.

```
NODE delete_item (int item, NODE head)
{
  NODE prev, cur, next;
  if (head->rlink == head)
  {
    printf ("List is empty\n");
    return head;
  }
  cur = head->rlink;
  while (cur != head)
  {
    if (item == cur->info) break;
    cur = cur->rlink;
  }
```

(7)

```
if (cur == head)
{
    printf ("Item not found \n");
    return head;
}
prev = cur -> llink;
next = cur -> rlink;
prev -> rlink = next;
next -> llink = prev;
free (cur);
return head;
}
```

Q.2: (9) /* Circular Queues Using Dynamic Array */

```
void Insert Q ()
{
    if (count == QUE_SIZE)
    {
        printf ("Queue is full : Increase Size \n");
        QUE_SIZE ++;
        REALLOC (q, QUE_SIZE, int);
    }
```

(4)

```
{
  for (i = QUE_SIZE - 2 ; i >= FRONT, i--)
  {
    q[i+1] = q[i];
  }
  FRONT++;
}
REAR = (REAR + 1) % QUE_SIZE;
q[REAR] = ITEM;
Count ++;
}
```

8.2(b) (i) /* Inorder Traversal */ — (6)

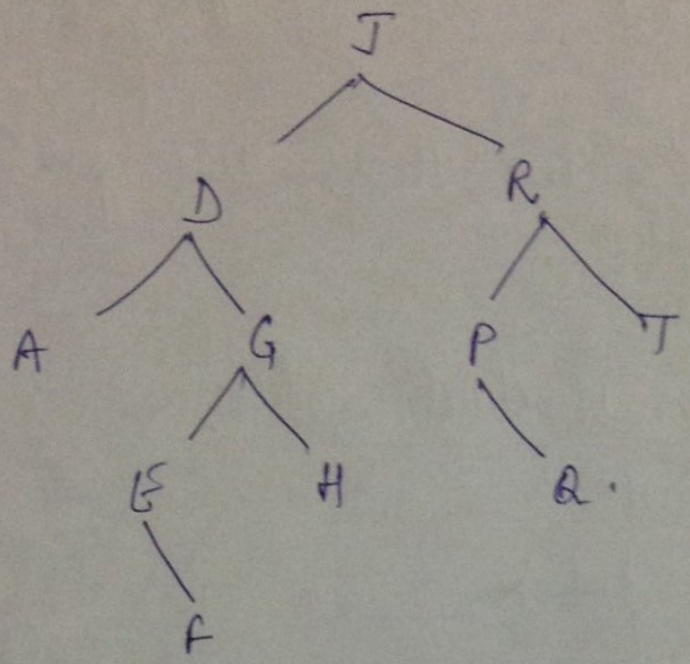
```
void inorder (NODE root)
{
  if (root == NULL) return;
  inorder (root -> llink);
  printf ("%d", root -> info);
  inorder (root -> rlink);
}
```

(ii) /* Preorder Traversal */

```
void preorder (NODE root)
{
  if (root == NULL) return;
  printf ("%d", root -> info);
  preorder (root -> llink);
  preorder (root -> rlink);
}
```

Delete M.

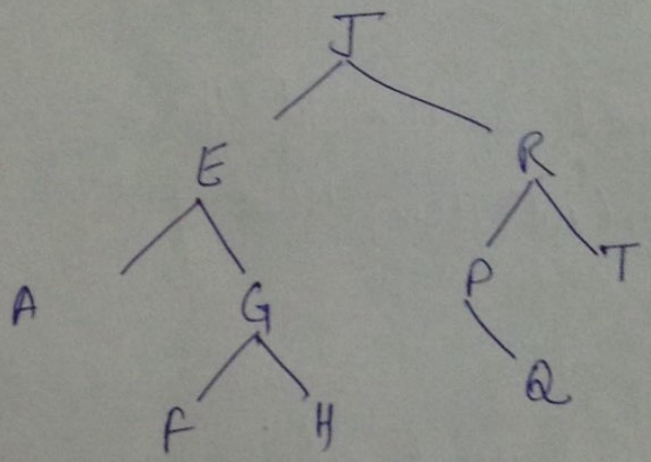
(ii)



—(2)

Inorder: A D E F G H J P Q R T

(iii) Delete D.



—(2)

Inorder: A E F G H J P Q R T.

Q4.

/* C func. to insert an item at the front end */

```
NODE insert_front (int item, NODE first)
{
  NODE temp;
  MALLOC (temp, 1, struct node);
  temp -> info = item;
  temp -> link = first;
  return temp;
}
```

(5)

/* C func. to insert an item at the rear end */

```
NODE insert_rear (int item, NODE first)
{
  NODE temp;
  NODE cur;
  MALLOC (temp, 1, struct node);
  temp -> info = item;
  temp -> link = NULL;
  if (first == NULL) return temp;
  cur = first;
  while (cur -> link != NULL)
  {
    cur = cur -> link;
  }
  cur -> link = temp;
  return first;
}
```

(5)

Q.5 (a) /* C func. to insert a node in a BST */.

```
NODE insert (int item, NODE root)
{
  NODE temp, cur, prev;
  MALLOC (temp, 1, struct node);
  temp->info = item;
  temp->llink = NULL;
  temp->rlink = NULL;
if (root == NULL) return temp;
  prev = NULL
  cur = root;
  while (cur != NULL)
   {
     prev = cur;
     if (item < cur->info)
        cur = cur->llink;
     else
        cur = cur->rlink;
   }
  if ( item < prev->info)
        prev->llink = temp;
   else
        prev->rlink = temp;
  return root;
}
```

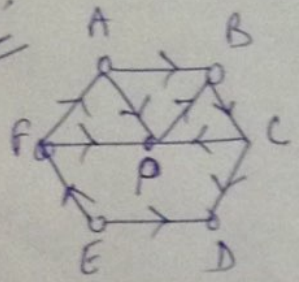
(5)

Q.5 (b) /* Breadth first Search */.

// Algo executes a BFS in Graph G, beginning at starting node A.

1. Initialise all nodes to the ready state (STATUS = 1)
2. Put the starting node A in QUEUE and change its status to the waiting state. (STATUS = 2).
3. Repeat steps 4 and 5 until QUEUE is empty.
4. Remove the front node N of QUEUE. Process N and change the status of N to the processed state (STATUS = 3).
5. Add to the rear of QUEUE all the neighbours of N that are in the Ready State (STATUS = 1). and change their status to the waiting state (STATUS = 2).
6. EXIT.

A to D:



Adj List:

A : P, B
B : C
C : P, D
D :
E : D, F
F : P, A
P : B

a) QUEUE : A — FRONT = 1, REAR = 1
ORIG : ϕ

b) QUEUE : ~~A~~, P, B — FRONT = 2, REAR = 3
ORIG : ϕ, A, A

c) QUEUE : ~~A~~, ~~P~~, B, — FRONT = 3, REAR = 3
ORIG : ϕ, A, A

d) QUEUE : ~~A~~, ~~P~~, ~~B~~, C — FRONT = 4, REAR = 4.
ORIG : ϕ, A, A, B

e) QUEUE : ~~A~~, ~~P~~, ~~B~~, C, D — FRONT = 5, REAR = 5
ORIG : ϕ, A, A, B, C

f) QUEUE : ~~A, P, B, C, D~~ — ~~FRONT = 6~~, ~~REAR =~~
A, P, B, C, D
ORIG

Path: D ← C ← B ← A

ve the algorithm for B... ace the algorithm and demonstrate how BFS works for ... oice.

Give the C implementation of **PUSH** and **POP** functions. Include suitab [
cks in the function.

Consider the following stack of characters, where STACK is allocated
mory cells. ITEM stores the element deleted from the STACK

STACK: A,C,D,F,K,___ , ___, ___

(For notational convenience, we use "___" to represent
pty memory cell.)

escribe the stack as the following operations take place:

POP(STACK,ITEM)
POP(STACK,ITEM)
PUSH(STACK,L)
PUSH(STACK,P)
POP(STACK,ITEM)
PUSH(STACK,R)
PUSH(STACK,S)
POP(STACK,ITEM) .

A, C, D, F, —, —, —, —
A, C, D, —, —, —, —, —
A, C, D, L, —, —, —, —
A, C, D, L, P, —, —, —
A, C, D, L, —, —, —, —
A, C, D, L, R, —, —, —
A, C, D, L, R, S, —, —
A, C, D, L, R, —, —, —

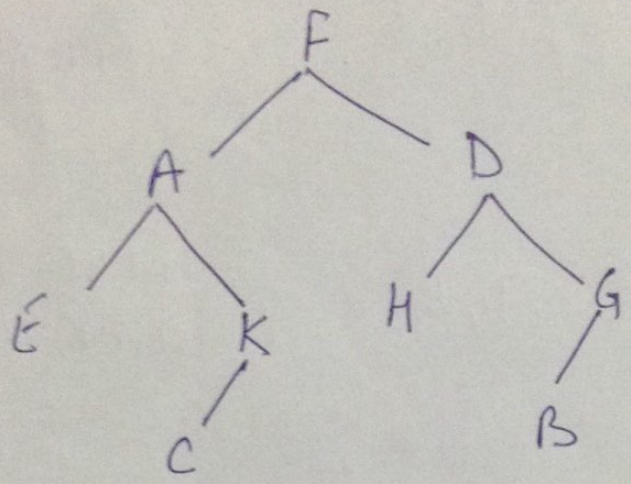
binary tree T has **9** nodes. The inorder and preorder traversals of T yield
llowing sequence of nodes:

Inorder: EACKFHDBG **Preorder: FAEKCDHGB**

aw the tree **T**

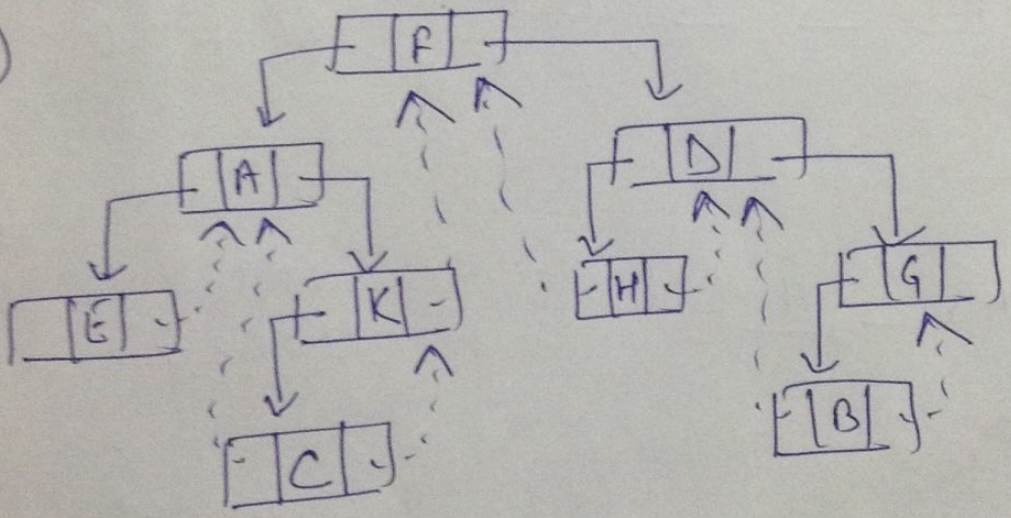
raw the inorder threading of the above binary tree T. Define the structure of
f this tree in C. [0

97(a). Inorder: E A C K F H D B G
Preorder: F A E K C D H G B.



4

(b)



8

Q.5 (b).

```
struct node
{
   int info;
   struct node *llink;
   struct node *rlink;
   int lthread;
   int rthread;
};
struct node *NODE;
```

(3)