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

| Sub: | DATA STRUCTURES AND APPLICATIONS | | | | | Sub Code: | BCS304 |
|---|---|---|---|---|---|---|---|
| Date: | | Duration: | **90 minutes** | Max Marks: | **50** | Sem/Sec:3 A,B,CSE-AIML | |

## SCHEME AND SOLUTIONS

1 a What is the disadvantage of the ordinary queue and how it is solved in a circular queue. Explain with a suitable example of how you would implement a circular queue.

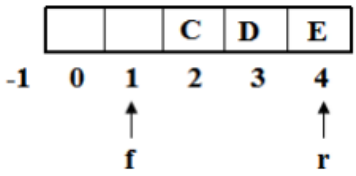
**Answer: -**

**Drawback -2M**

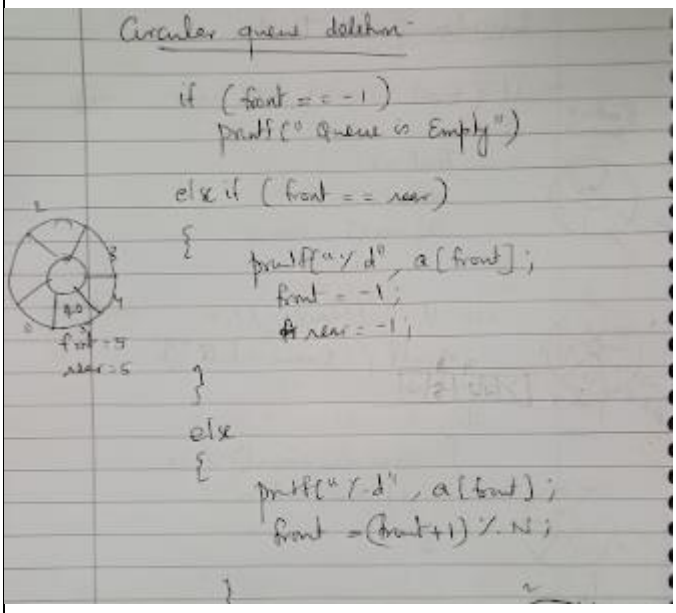
**Circular Queue Implementation-either pictorially or through pseudo code 4M**

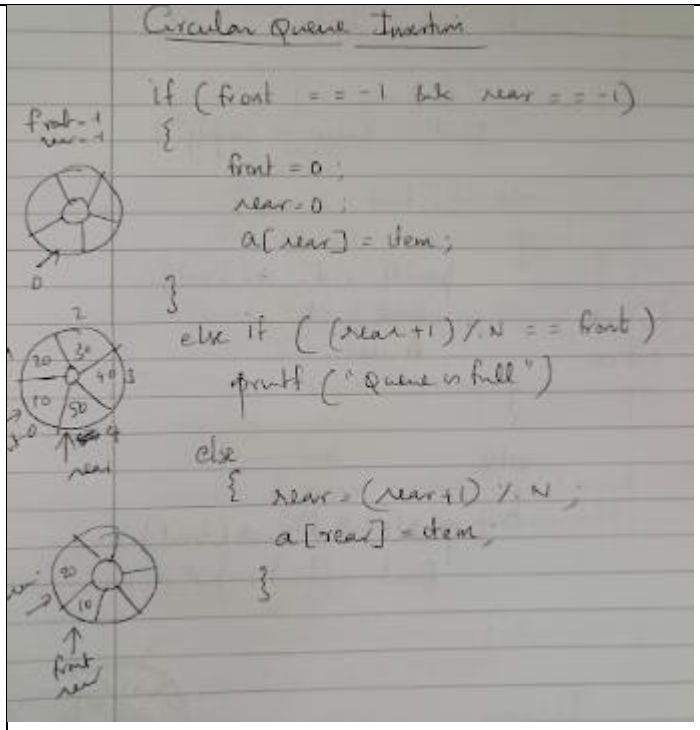
**Drawback of Queue**

When item enters and deleted from the queue, the queue gradually shifts to the right as shown in figure.




In this above situation, when we try to insert another item, which shows that the **queue is full**. This means that the **rear** index equals to MAX_QUEUE_SIZE -1. But even if the space is available at the front end, rear insertion cannot be done.






b Differentiate between arrays and linked list.

**Answer: -**

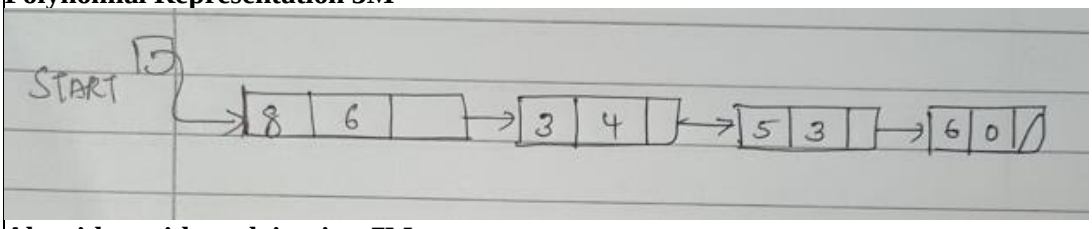
**Any 4 differences-1X4=4**

| ARRAY | LINKED LISTS |
|---|---|
| 1. Arrays are stored in contiguous location. | 1. Linked lists are not stored in contiguous location. |
| 2. Fixed in size. | 2. Dynamic in size. |
| 3. Memory is allocated at compile time. | 3. Memory is allocated at run time. |
| 4. Uses less memory than linked lists. | 4. Uses more memory because it stores both data and the address of next node. |
| 5. Elements can be accessed easily. | 5. Element accessing requires the traversal of whole linked list. |
| 6. Insertion and deletion operation takes time. | 6. Insertion and deletion operation is faster. |

2 a Represent the polynomial $8x^6+3x^4+5x^3+6$ using singly linked list. Write a function to perform addition to two polynomials with single variable using linked list.

**Answer: -**

**Polynomial Representation 3M**



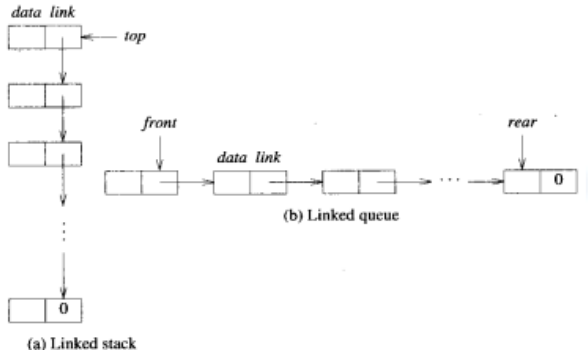

**Algorithm with explaination-7M**

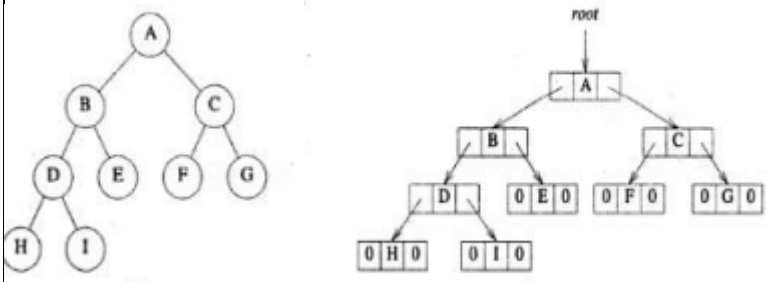
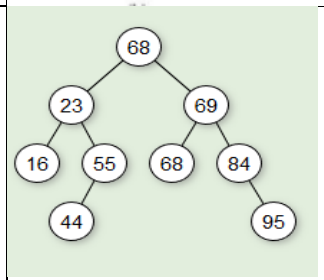
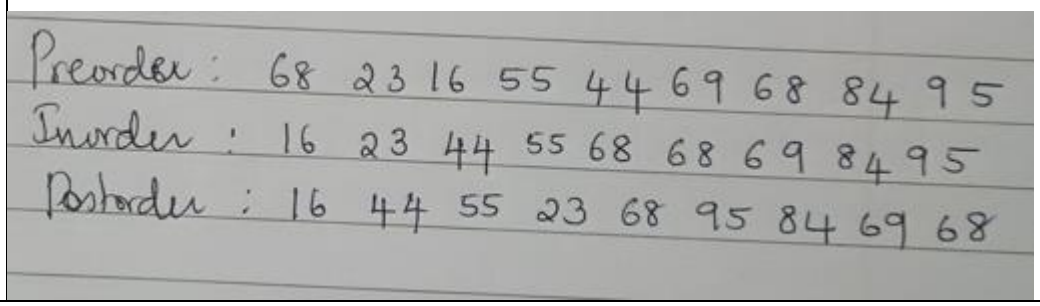
```
polyPointer padd(polyPointer a, polyPointer b)
{/* return a polynomial which is the sum of a and b */
  polyPointer c, rear, temp;
  int sum;
  MALLOC(rear, sizeof(*rear));
  c = rear;
  while (a && b)
    switch (COMPARE(a→expon,b→expon)) {
      case -1: /* a→expon < b→expon */
           attach(b→coef,b→expon,&rear);
           b = b→link;
           break;
      case 0: /* a→expon = b→expon */
           sum = a→coef + b→coef;
           if (sum) attach(sum,a→expon,&rear);
           a = a→link;  b = b→link; break;
      case 1: /* a→expon > b→expon */
           attach(a→coef,a→expon,&rear);
           a = a→link;
    }
  /* copy rest of list a and then list b */
  for (; a; a = a→link) attach(a→coef,a→expon,&rear);
  for (; b; b = b→link) attach(b→coef,b→expon,&rear);
  rear→link = NULL;
  /* delete extra initial node */
  temp = c; c = c→link;  free(temp);
  return c;
}
```

| | | |
|---|---|---|
| 3 | a | Develop C functions to perform following operations using Singly linked list:<br>**Answer: -**<br><br>i) Create three node list with data 50, 80 and 60 -**2M**<br>Logic for:-<br>Node creation<br>Assigning values and links<br><br>ii) Delete the node whose data value is 60-**2M**<br>Logic of Delete the last node from the list<br><br>iii) Insert the node whose data value is 40 at the beginning-**2M**<br><br>Logic of inserting node at the beginning to be wrritten |
| | b | Explain how linked list can be used to implement stack and queue.<br>**Answer: -**<br><br>**Concept Explanation -2X2=4M** |

## LINKED STACKS AND QUEUES

The below figure shows stacks and queues using linked list. Nodes can easily add or delete a node from the top of the stack. Nodes can easily add a node to the rear of the queue and add or delete a node at the front




| | | |
|---|---|---|
| 4 | a | Explain: 1) Binary tree 2) Complete Binary Tree 4) Skewed Binary Tree<br>**Answer: -**<br><br>**Binary Tree-2M with diagram**<br>This is a finite set of nodes that is either empty or consists of → a root & → two disjoint binary trees called left subtrees and right subtrees<br>**Complete binary Tree-2M with diagram**<br>Complete tree is a binary tree in which every level except possibly last level is completely filled. A binary tree with n nodes & depth k is complete iff its nodes correspond to nodes numbered from 1 to n in full binary tree of depth k<br>**Skewed binary Tree-2M with diagram**<br>Skewed tree is a tree consisting of only left subtree or only right subtree |
| | b | Explain the linked list representation of a binary tree by taking a suitable example<br>**Answer: -**<br><br>**Any binary tree with linked list representation-4M**<br>  <br> |
| 5 | a |  <br><br>Write the preorder, inorder, postorder traversal of the binary tree given in the figure with C code snippets.<br>**Answer: -**<br><br>**Traversal Preorder and Inorder- 3M**<br>**C functions for Preorder and Inorder -3M**<br>  <br> |

```
void preorder(tree_pointer ptr)
{       /* preorder tree traversal */
        if (ptr)
        {
                printf("%d", ptr->data);
                preorder(ptr->left_child);
                predorder(ptr->right_child);
         }
}
```

```
void inorder(tree_pointer ptr)
{       /* inorder tree traversal */
        if (ptr)
        {
              inorder(ptr->left_child);
              printf("%d", ptr->data);
              indorder(ptr->right_child);
        }
}
```

Program 5.1: Inorder traversal of binary tress

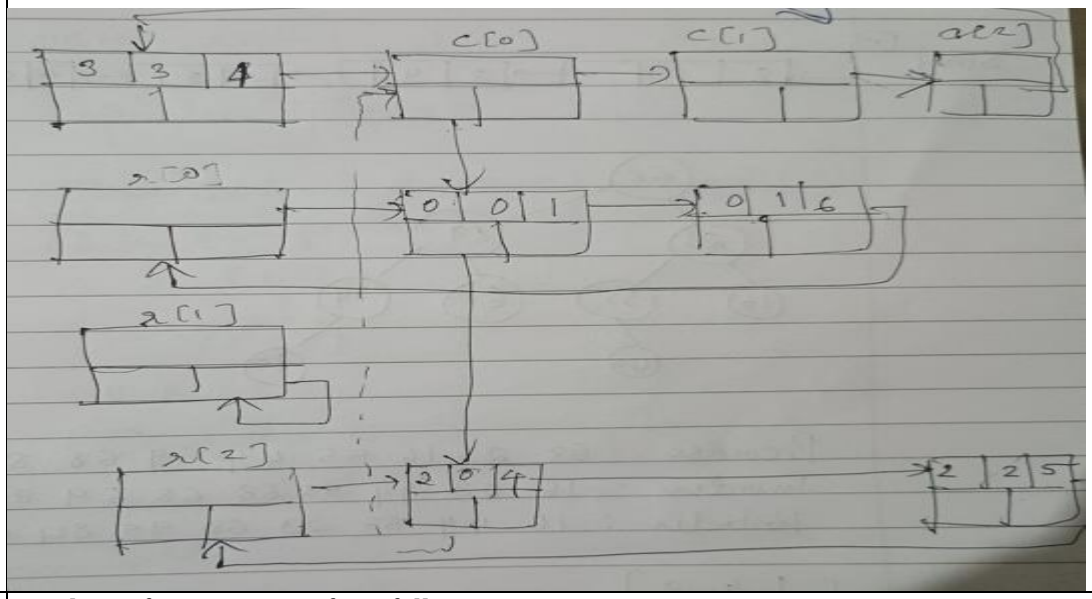
b

$$\begin{bmatrix} 1 & 6 & 0 \\ 0 & 0 & 0 \\ 4 & 0 & 5 \end{bmatrix}$$

Represent the given sparse matrix using linked list.

**Diagram- 4M**

**Answer: -**



6 a

Develop C functions to perform following operations:

i) To concatenate two singly linked list

ii) Insert and delete a node from the beginning and end of a circular doubly linked list

**Answer: -**

**C function 2X3=6M**

**Logic to Concatenate two linked list 3M**

| | | |
|---|---|---|
| | | List 1: Start List2: Start 2<br><br>Logic to Find the end of List 1 and last node of List 1 is assigned with Start2<br><br>**Insert and delete from beginning and end of circular doubly linked list 3M**<br><br>Logic to insert and delete node from beginning and end of circular doubly linked list. |
| | b | Differentiate between circular doubly linked list and doubly linked list.<br>**Answer: -**<br><br>  <br><br>**Difference 2X2=4M** |

| Doubly Linked Lists (DLL) | Circular Linked List (CLL) |
|---|---|
| struct Node<br>{<br>int Data;<br>struct Node *Next;<br>struct Node<br>*Previous;<br>}; | Depends on the type of circular linked listening<br>**Single Circular:**<br>struct Node<br>{<br>int Data;<br>struct Node *Next;<br>};<br>**Double Circular:**<br>struct Node<br>{<br>int Data;<br>struct Node *Next;<br>struct Node *Previous;<br>}; |
| Pointers contains the address of next node as well as previous Node in the list. | Pointer can or cannot contains the address of previous node as it depends on type of circular linked list. |

**CI** **CCI** **HOD**

----------All the Best----------

| **CO-PO Mapping** | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Course Outcomes** | | **Modules covered** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** | **PSO4** |
| CO1 | Explain different data structures and their applications. | 1 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | 2 | 3 | 2 | 2 | - |
| CO2 | Apply Arrays, Stacks and Queue data structures to solve the given problems. | 2 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | 2 | 3 | 2 | 2 | - |
| CO3 | Use the concept of linked list in problem solving. | 3 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | 2 | 3 | 2 | 2 | - |
| CO4 | Develop solutions using trees and graphs to model the real-world problem | 4 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | 2 | 3 | 2 | 2 | - |
| CO5 | Explain the advanced Data Structures concepts such as Hashing Techniques and Optimal Binary Search Trees | 5 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | 2 | 3 | 2 | 2 | - |