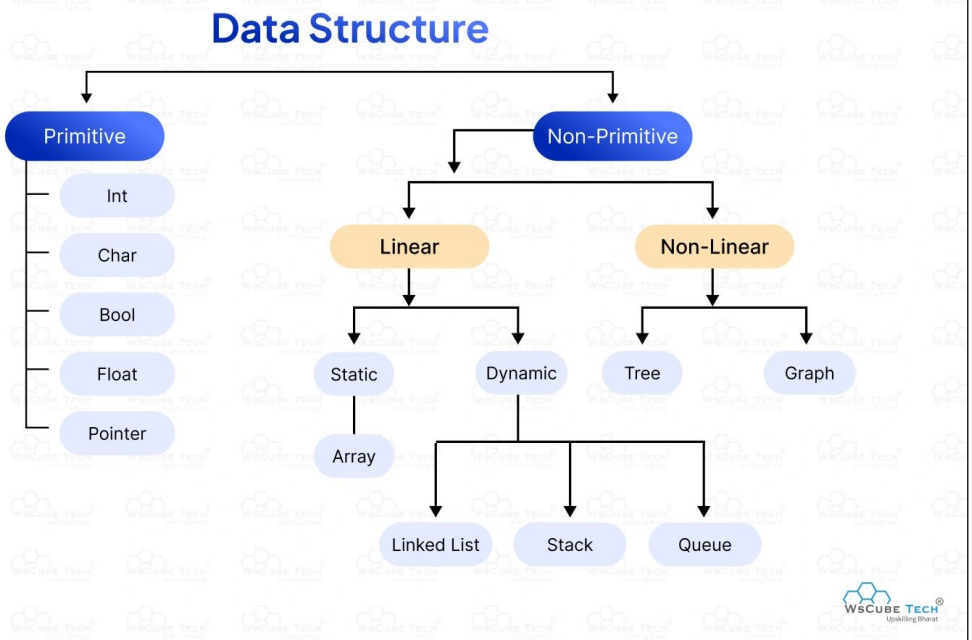
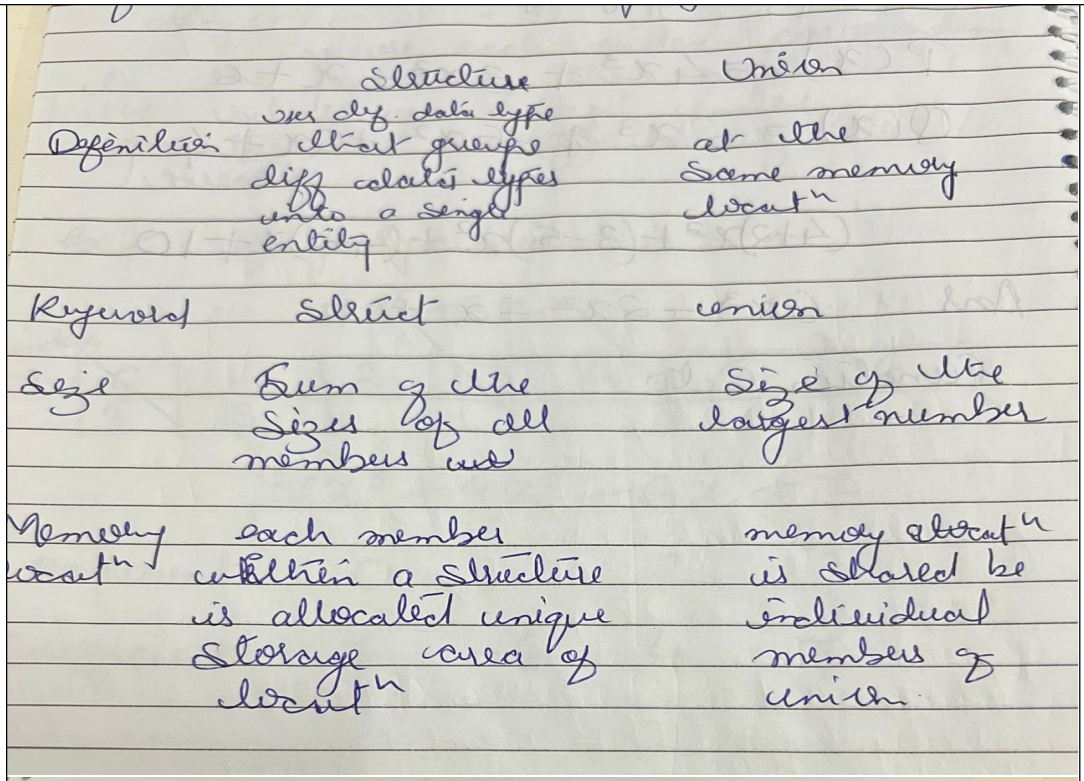
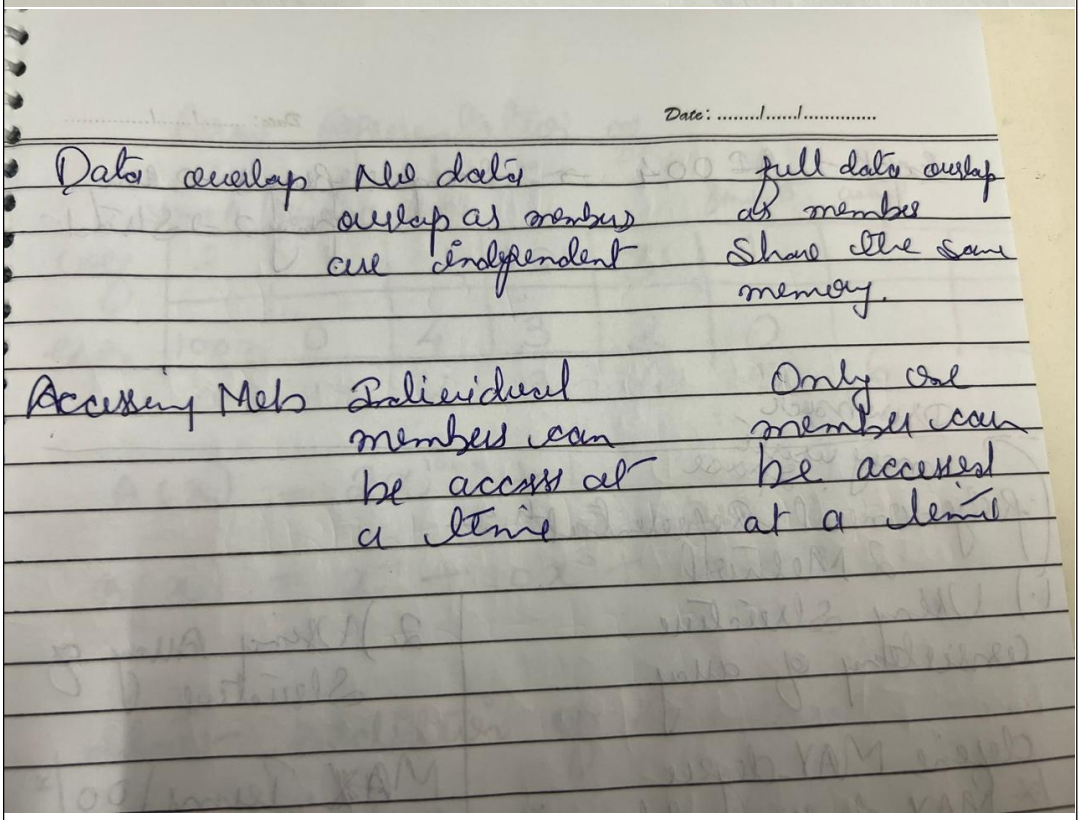
USN | | | | | | | | | | |

Internal Assessment Test 1 –November-2025

| Sub: | DATA STRUCTURES AND APPLICATIONS | | | | | Sub Code: | BCS304 | Branch: | AIML & CSML |
|---|---|---|---|---|---|---|---|---|---|
| Date: | | Duration: | 90 minutes | Max Marks: | 50 | Sem/Sec: | III -A, B, C | | OBE |

| | | Answer any FIVE FULL Questions | MARKS | CO | RBT |
|---|---|---|---|---|---|
| 1 | a | Define Data Structures. Explain the classification o f data structures with a neat diagram.<br>DATA STRUCTURE**1M**<br>Data structure is a representation of the logical relationships existing between individual elements of data. A data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.<br>The logical or mathematical model of a particular organization of data is called a data structure.<br><br>The non-primitive data structures is further classified into **1M(daigram)**<br>  <br><br>1. Linear Data Structure 2**M**<br>2. Non-linear Data Structure **2M**<br>1. Linear Data Structure: A data structure is said to be linear if its elements form a sequence or a linear list. There are basically two ways of representing such linear structure in memory. a. One way is to have the linear relationships between the elements represented by means of sequential memory location. These linear structures are called arrays.<br>b. The other way is to have the linear relationship between the elements represented by means of pointers or links. These linear structures are called linked lists. The common examples of linear data structure are Arrays, Queues, Stacks, Linked lists<br>2. Non-linear Data Structure: A data structure is said to be non-linear if the data are not arranged in sequence or a linear. The insertion and deletion of data is not possible in linear fashion. This structure is mainly used to represent data containing a hierarchical relationship between elements. Trees and graphs are the examples of non-linear data structure. | 6 | 1 | L2 |
| | b | Differentiate structures and unions.<br><br>**Solution**<br>**3M** | 4 | 1 | L2 |

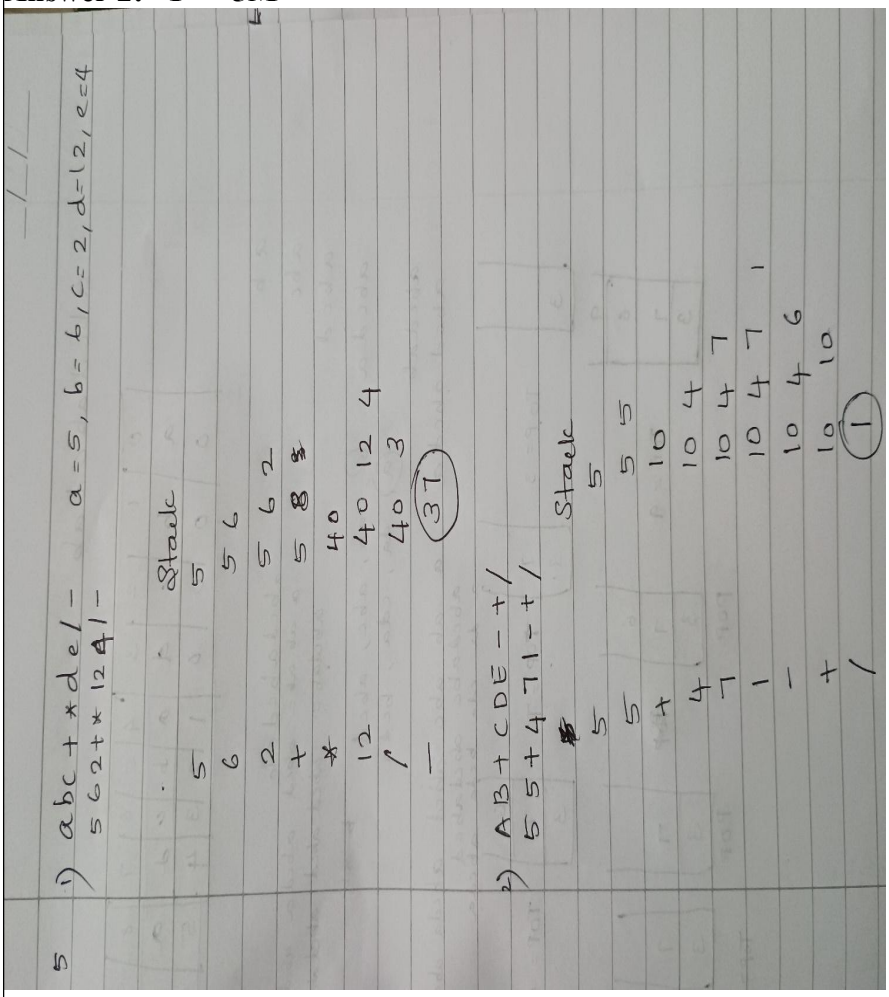
| | | | | | |
|---|---|---|---|---|---|
| | |  <br><br>  <br><br>Example Declaration **1M**<br>structure<br>struct student { int id; char name[20]; float marks; };<br>union<br>union data { int i; float f; char ch; }; | | | |
| 2 | a | **Explain various memory allocation techniques? Explain how memory can be dynamically allocated using malloc().**<br>Solution<br>various memory allocation techniques<br>**Static Memory Allocation 1M**<br>Static Memory is allocated for declared variables by the compiler. The address can be found using the address *of* operator and can be assigned to a pointer. The memory is allocated during | 4 | 2 | L2 |

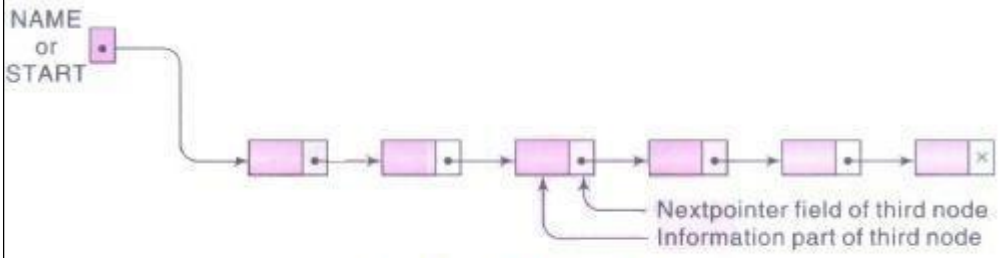
| | | | | | |
|---|---|---|---|---|---|
| | | compile time.<br><br>**Memory allocation 1M**<br><br>Memory allocation done at the time of execution(run time) is known as dynamic memory allocation. Functions calloc() and malloc() support allocating dynamic memory. In the Dynamic allocation of memory space is allocated by using these functions when the value is returned by functions and assigned to pointer variables.<br><br>Dynamic Memory Allocation functions in C: **2M**<br>malloc( )<br><br>Related header file is stdlib.h><br>These functions provide a complete set of memory allocation, reallocation, deallocation, and heap management tools.<br>malloc()<br>Syntax:<br><br>ptr = (cast-type*) malloc(byte-size);<br><br>Here, ptr is pointer of cast-type. The malloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.<br>Example:<br><br>ptr = (int*) malloc(100 * sizeof(int)); | | | |
| | b | **Write a C Functions to implement pop, push and display operations for stacks using arrays.**<br>2(each function)*3=6<br>**Solution:- 2(each function)*3=6**<br>int stack[MAX];<br>int top = -1;<br><br>// Function to push an element into the stack<br>void push(int value) {<br>if (top == MAX - 1) {<br>printf("Stack Overflow! Cannot push %d\n", value);<br>} else {<br>top++;<br>stack[top] = value;<br>printf("%d pushed into the stack.\n", value);<br>}<br>}<br><br>// Function to pop an element from the stack<br>void pop() {<br>if (top == -1) {<br>printf("Stack Underflow! No elements to pop.\n");<br>} else {<br>printf("%d popped from the stack.\n", stack[top]);<br>top--;<br>}<br>}<br><br>// Function to display all elements in the stack<br>void display() {<br>if (top == -1) {<br>printf("Stack is empty.\n");<br>} else {<br>printf("Stack elements are:\n");<br>for (int i = top; i >= 0; i--) {<br>printf("%d\n", stack[i]);<br>}<br>}<br>} | 6 | 1 | L3 |
| 3 | | Evaluate the following postfix expressions,<br>1. abc+*de/- where a=5, b=6, c=2, d=12, e=4<br>2. AB+CDE-+/ where A=5, B=5, C=4, D=7, E=1 | 10 | 2 | L3 |

Solution

**Answer 1: 37 5M**

**Answer 2: 1 5M**




| | | | | | |
|---|---|---|---|---|---|
| 4 | | Explain linked singly linked list with example:<br>i) Insert a node o f the beginningii)Delete a node at the front iii) Display<br><br>Solution -<br>singly linked list explaination **2.5M**<br>-A linked list is a dynamic data structure where each element (called a node) is made up of two items - the data and a reference (or pointer) which points to the next node. A linked list is a collection of nodes where each node is connected to the next node through a pointer.<br>  <br><br>Fig: Linked list with 6 nodes<br>The following examples of linked lists indicate that the nodes of a list need not occupy adjacent elements in the arrays INFO and LINK, and that more than one list may be maintained in the same linear arrays INFO and LINK. However, each list must have its own pointer variable giving the location of its first node.<br><br>START=9 INFO[9]=N<br>LINK[3]=6 INFO[6]=V<br>LINK[6]=11 INFO[11]=E<br>LINK[11]=7 INFO[7]= X<br>LINK[7]=10 INFO[10]= I<br>LINK[10]=4 INFO[4]= T<br>LINK[4]= NULL value, So the list has ended | 10 | 3 | L2 |

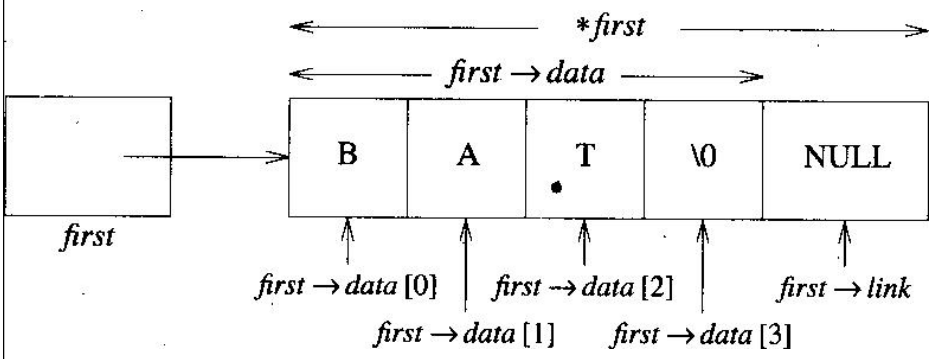
Defining a node structure

typedef struct listNode *listPointer typedef struct {
char data[4]; listPointer list;
} listNode;

Create a New Empty list
listPointer first = NULL

To create a New Node
MALLOC (first, sizeof(*first));

To place the data into NODE
strcpy(first→ data,"BAT"); first→ link = NULL




**Function to insert a node at the beginning-2.5M**

```
void insertAtBeginning(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = head; // Point new node to current head
   head = newNode;       // Move head to new node
   printf("%d inserted at the beginning.\n", value);
}
```

**Function to delete a node from the front-2.5M**

```
void deleteFromFront() {
   if (head == NULL) {
      printf("List is empty. Nothing to delete.\n");
   } else {
      struct Node* temp = head;
      head = head->next; // Move head to next node
      printf("%d deleted from the front.\n", temp->data);
      free(temp); // Free old head
```

**Function to display the list- 2.5M**

```
void display() {
   if (head == NULL) {
      printf("List is empty.\n");
   } else {
      struct Node* temp = head;
      printf("Linked List: ");
      while (temp != NULL) {
         printf("%d -> ", temp->data);
         temp = temp->next;
      }
```

```
        printf("NULL\n");
    }
}
```

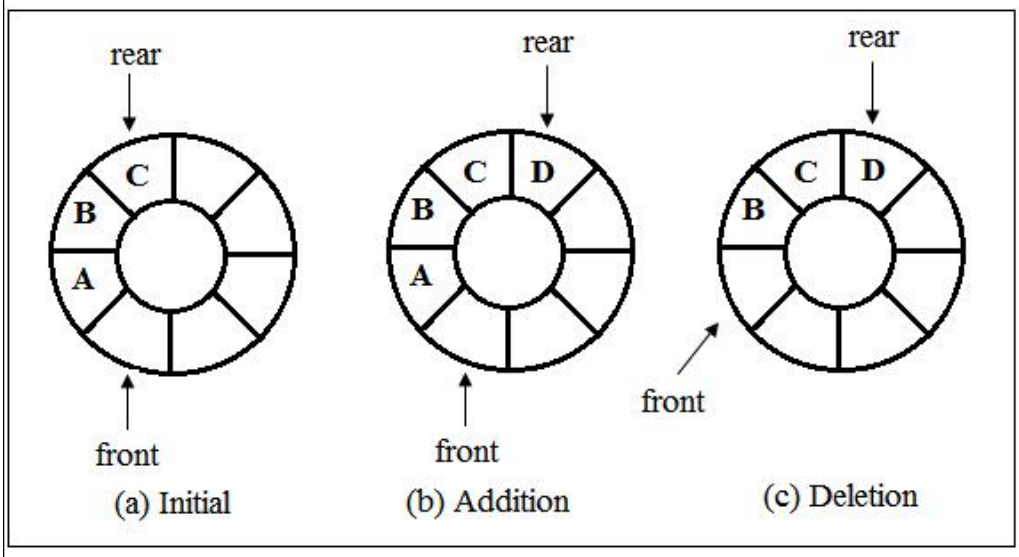
5

What is Ciculay queue? How insertion () , deletion () and display ()done on circular queue.

**CIRCULAR QUEUES- 2.5**

It is "The queue which **wrap around** the end of the array." The array positions are arranged in a circle as shown in figure.

In this convention the variable **front** is changed. **front** variable points one position **counterclockwise** from the location of the front element in the queue. The convention for rear is unchanged.




**Implementation of Circular Queue Operations**

When the array is viewed as a circle, each array position has a next and a previous position. The position next to MAX-QUEUE-SIZE -1 is 0, and the position that precedes 0 is MAX-QUEUE-SIZE -1.

10 | 2 | L2

When the queue rear is at MAX_QUEUE_SIZE-1, the next element is inserted at position 0.

In circular queue, the variables front and rear are moved from their current position to the next position in clockwise direction. This may be done using code

```
if (rear = = MAX_QUEUE_SIZE-1) rear = 0;
else rear++;
```

**Add to a circular queue 2.5M**

```
void addq(element item)
{
rear = (rear +1) % MAX_QUEUE_SIZE; if (front == rear)
queueFull();
queue [rear] = item;
}
```

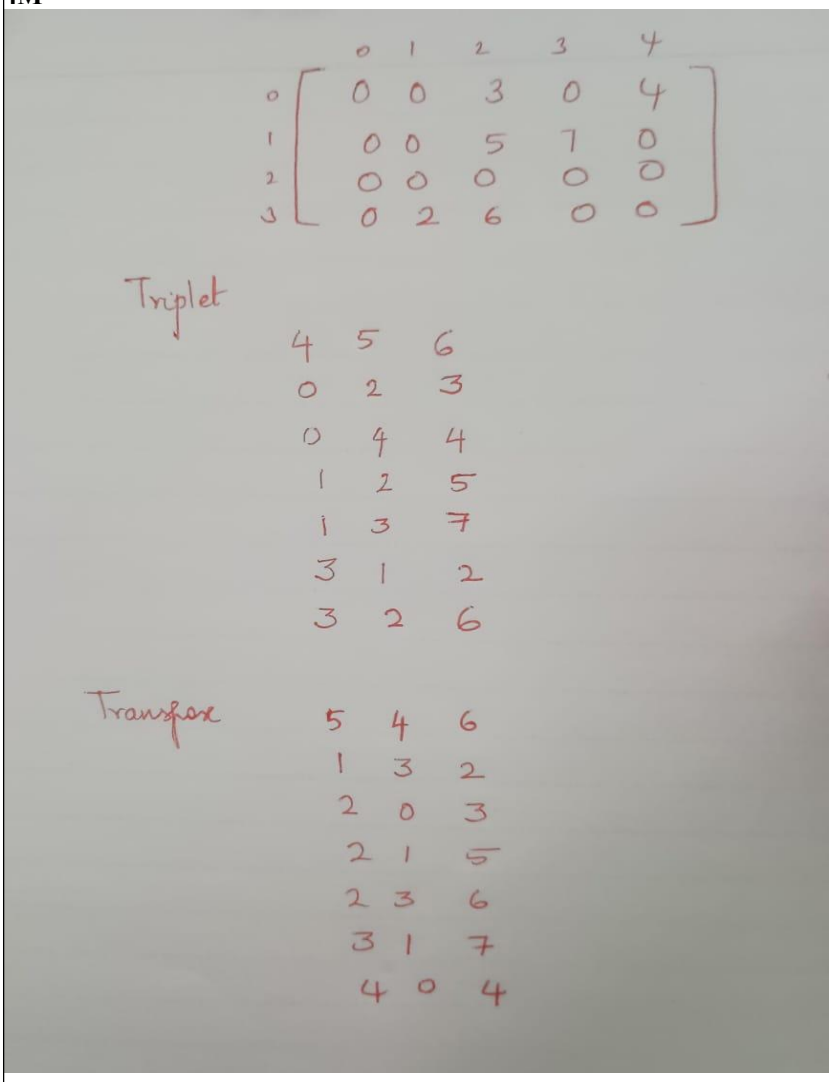
**Delete from a circular queue 2.5M**

```
element deleteq()
{ /* remove front element from the queue */ element item;
if (front == rear)
return queueEmpty( ); /* return an error key */ front = (front+1)% MAX_QUEUE_SIZE;
return queue[front];
}
```

## Display 2.5M

Steps:

If queue is empty, display message.

| | | | | | |
|---|---|---|---|---|---|
| | | Start from front, print elements until you reach rear.<br>Use (i + 1) % MAX to move circularly. | | | |
| 6 | | $A = \begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$<br>Discuss Sparse matrix? Give the triplet form o f a given matrix and find its transpose<br>SPARSE MATRIX REPRESENTATION **2M**<br>• We can classify uniquely any element within a matrix by using the triple . Therefore, we can use an array of triples to represent a sparse matrix.<br>• Sparse matrix contains many zero entries.<br>• When a sparse matrix is represented as a 2-dimensional array, we waste space For ex, if 100*100 matrix contains only 100 entries then we waste 9900 out<br><br>(a)<br><br>(b)<br><br>**4M**<br>  | 10 | 3 | L3 |