EX. NO :3B **QUEUE ADT USING LINKED LIST**

AIM:

To write a C program for Queue using Linked implementation.

DESCRIPTION:

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data(enqueue) and the other is used to remove data(dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.

Queue operations may involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory. Here we shall try to understand the basic operations associated with queues.

- enqueue () − add (store) an item to the queue.
- dequeue () − remove (access) an item from the queue.

Few more functions are required to make the above-mentioned queue operation efficient. These are

- peek() − Gets the element at the front of the queue without removing it.
- isfull() − Checks if the queue is full.
- isempty() − Checks if the queue is empty.

ALGORITHM:

1. Define a struct for each node in the queue. Each node in the queue

   contains data and link to the next node. Front and rear pointer points to first and last node inserted in the queue.

2. The operations on the queue are

    a. INSERT data into the queue
    b. DELETE data out of queue

3. INSERT DATA INTO queue

    a. Enter the data to be inserted into queue.

    b. If TOP is NULL

        i. The input data is the first node in queue. ii.
        The link of the node is NULL. iii. TOP
        points to that node. c. If
    TOP is NOT NULL
        i. The link of TOP points to the new node. ii.
        TOP points to that node.

4. DELETE DATA FROM queue a. If
    TOP is NULL
        i. the queue is empty b. If
    TOP is NOT NULL
        i. The link of TOP is the current TOP.
ii. The pervious TOP is popped from queue.

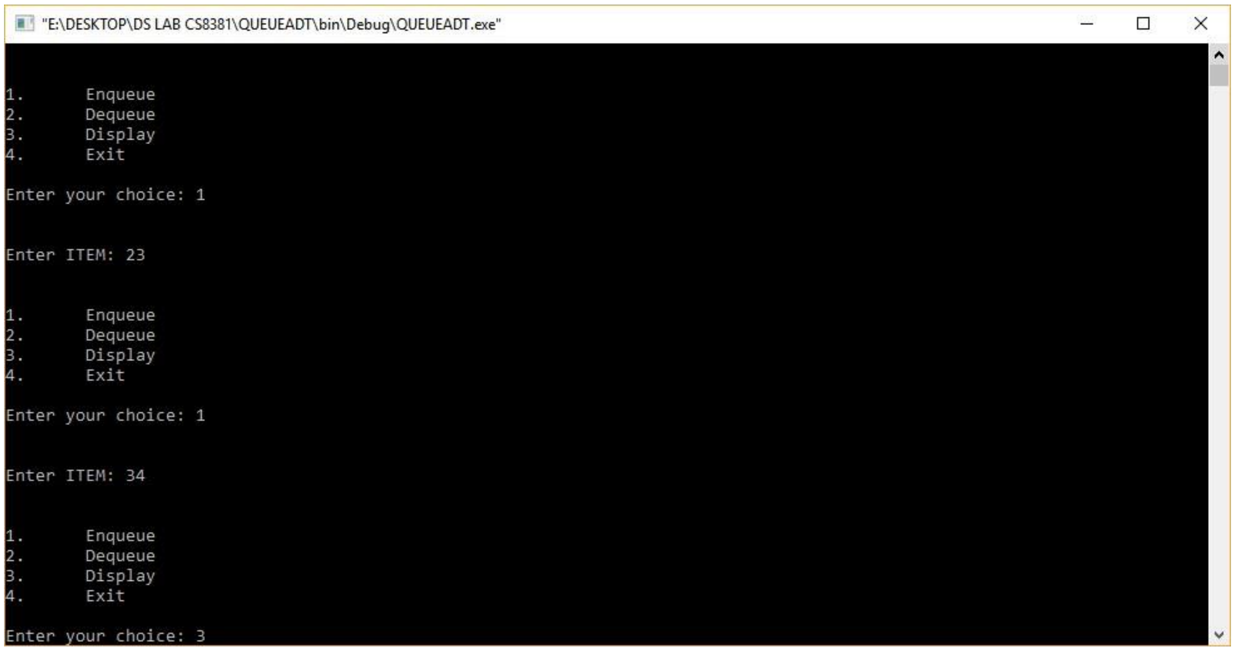
5. The queue represented by linked list is traversed to display its content.

PROGRAM:

```
#include<stdio.h>

#include<conio.h>

struct node { int

info; struct node

*link;

}*front = NULL, *rear = NULL;

 void insert();

void delet(); void
```

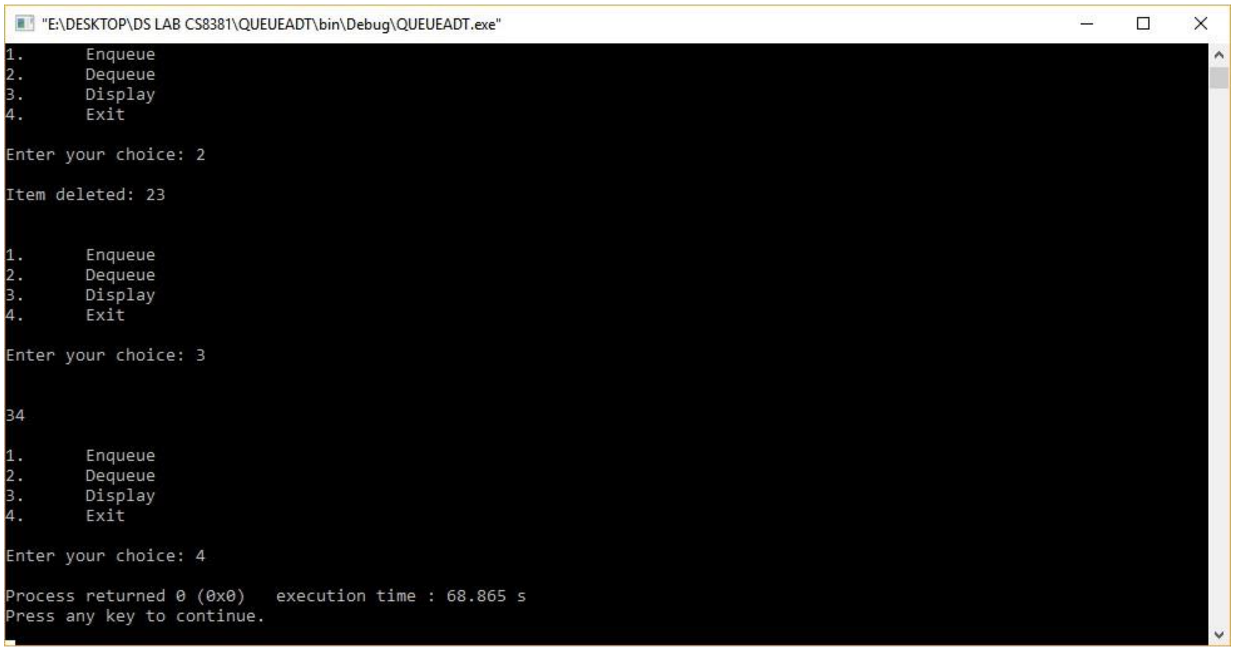
```
display(); int
item; void main()
{  int ch; do {
printf("\n\n1.\tEnqueue\n2.\tDequeue\n3.\tDisplay\n4.\tExit\n");
printf("\nEnter your choice: "); scanf("%d", &ch); switch(ch) {
case 1: insert(); break; case 2: delet(); break; case 3: display();
break; case 4: exit(0); default:
printf("\n\nInvalid choice. Please try again...\n");
} } while(1);
getch();
}  void insert() {
printf("\n\nEnter ITEM: ");
scanf("%d", &item); if(rear ==
NULL)
{ rear = (struct node *)malloc(sizeof(struct node));
rear->info = item; rear->link = NULL; front = rear; }
else{
 rear->link = (struct node *)malloc(sizeof(struct node));
rear = rear->link; rear->info = item; rear->link = NULL;
}} void delet(){ struct node *ptr;
if(front == NULL) printf("\n\nQueue is
empty.\n"); else{ ptr = front; item =
front->info; front = front->link;
free(ptr); printf("\nItem deleted:
%d\n", item); if(front == NULL) rear =
NULL; }} void display() { struct node
*ptr = front; if(rear == NULL)
printf("\n\nQueue is empty.\n"); else {
printf("\n\n"); while(ptr != NULL)
{ printf("%d\t",ptr->info);
ptr = ptr->link;
}}}
```

OUTPUT



"E:\DESKTOP\DS LAB CS8381\QUEUEADT\bin\Debug\QUEUEADT.exe"

```
1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 1


Enter ITEM: 23


1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 1


Enter ITEM: 34


1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 3
```



"E:\DESKTOP\DS LAB CS8381\QUEUEADT\bin\Debug\QUEUEADT.exe"

```
1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 2

Item deleted: 23


1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 3


34

1.      Enqueue
2.      Dequeue
3.      Display
4.      Exit

Enter your choice: 4

Process returned 0 (0x0)   execution time : 68.865 s
Press any key to continue.
```

RESULT:

Thus the C program for array implementation of Queue ADT was created, executed and output was verified successfully