Experiment No. 4: Simple Queue Operations

Aim: To implement a Linear Queue using arrays.

Objective:

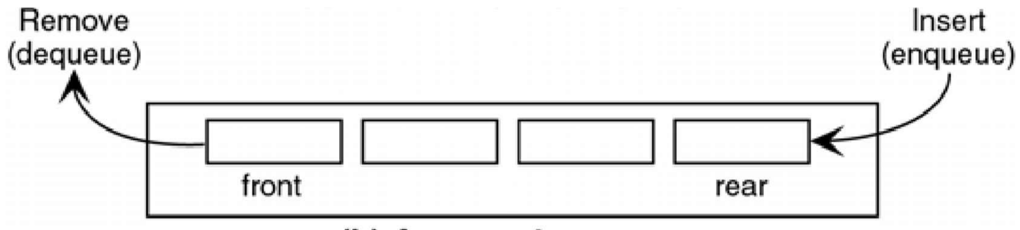
1 Understand the Queue data structure and its basic operations.

2. Understand the method of defining Queue ADT and its basic operations.

3. Learn how to create objects from an ADT and member functions are invoked.

Theory:

A queue is an ordered collection where items are removed from the front and inserted at the rear, following the First-In-First-Out (FIFO) order. The fundamental operations for a queue are "Enqueue," which adds an item to the rear, and "Dequeue," which removes an item from the front.



(b) A computer queue

Typically, a one-dimensional array is used to implement a queue, and two integer values, FRONT and REAR, track the front and rear positions in the array. When an element is removed from the queue, FRONT is incremented by one, and when an element is added to the queue, REAR is increased by one. This ensures that items are processed in the order they were added, maintaining the FIFO principle.

Algorithm:

```
ENQUEUE(item)
1. If (queue is full)
     Print "overflow"
2. if (First node insertion)
      Front++
3. rear++
Queue[rear]=value
DEQUEUE()
1. If (queue is empty)
```

```
    Print "underflow"
2.  if(front=rear)
        Front=-1 and rear=-1
3.  t = queue[front]
4.  front++
5.  Return t
```

ISEMPTY()

```
1. If(front = -1)then
        return 1
2. return 0
```

ISFULL()

```
1. If(rear = max)then
        return 1
2. return 0
```

Code:

```
 #include<stdio.h>
#include<stdlib.h>
#define maxsize 5
void insert();
void delete();
void display();
int front = -1, rear = -1;
int queue[maxsize];
void main () {
int choice;
while(choice != 4) {
```

```
printf("\n*************************Main Menu*****************************\n");
printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
printf("\nEnter your choice ?");
scanf("%d",&choice);
switch(choice) {
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nEnter valid choice??\n");
} }
void insert() {
int item;
printf("\nEnter the element\n");
scanf("\n%d",&item);
if(rear == maxsize-1) {
printf("\nOVERFLOW\n");
return;
}
if(front == -1 && rear == -1) {
front = 0;
rear = 0;
} else {
rear = rear+1;
}
queue[rear] = item;
printf("\nValue inserted ");
}
void delete() {
int item;
if (front == -1 || front > rear) {
printf("\nUNDERFLOW\n");
return;
} else {
item = queue[front];
if(front == rear) {
front = -1;
rear = -1 ;
} else {
front = front + 1;
}
printf("\nvalue deleted ");
}}
```

```
void display() {
int i;
if(rear == -1) {
printf("\nEmpty queue\n");
}else{
 printf("\nprinting values .....\n");
for(i=front;i<=rear;i++) {
printf("\n%d\n",queue[i]);
}} }
```

Output:

```
 *************Main Menu**************
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?1
Enter the element
123
Value inserted
*************Main Menu**************
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?1
Enter the element
```

```
90
Value inserted
*************Main Menu**************
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
value deleted
*************Main Menu**************
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?3
printing values .....
90
*************Main Menu**************
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?4
```

Conclusion:

What is the structure of queue ADT?

- A queue ADT (abstract data type) is a data structure that follows the FIFO (first in, first out) principle. It means that the element that is inserted first in the queue is the one that is removed first. A queue ADT has two main operations: enqueue and dequeue. Enqueue adds an element to the rear of the queue, and dequeue removes an element from the front of the queue. A queue ADT can also have other operations, such as peek, which returns the element at the front of the queue without removing it, or isEmpty, which checks if the queue is empty.

List various applications of queues?

- Queues are used to manage data flow and handle tasks in various applications, such as operating systems, network protocols, and data processing systems.
- Queues are used to implement algorithms like breadth-first search, which involves exploring nodes in a graph level-by-level.
- Queues are used to schedule tasks based on priority or the order in which they were received

Where is queue used in a computer system proceesing?

- One possible use of queue in computer system processing is to manage data flow and handle tasks in various applications, such as operating systems, network protocols, and data processing systems. Queues can help to store and process data in the order that they arrive, and to balance the load among different servers or devices.