Batch: A3Roll No.: 16010121051

Experiment / assignment / tutorial No.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Basic operations on queue for the assigned application using Array and Linked List- Create, Insert, Delete, Destroy

Objective: To implement Basic Operations on Queue i.e. Create, Push, Pop, Destroy for the given application

Expected Outcome of Experiment:

CO	Outcome
1	Explain the different data structures used in problem solving

Books/ Journals/ Websites referred:

1. *Fundamentals Of Data Structures In C* – Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
3. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg & Behrouz A. Forouzan
4.

Abstract:
(Define Queue, enlist queue operations).
Abstract:

(Define Queue, enlist queue operations).

• A queue in the data structure can be considered similar to the queue in

the realworld.

• Queues have a FIFO (first-in-first-out) structure, where deletion and insertion

happen at opposite ends.

• Queues have 2 pointers looking to the front and back of the queue. • Queues

use enqueue() and dequeue() functions : adding to the end of the queue and

removing from the front of the queue.

There are two fundamental operations performed on a Queue - enqueue and

dequeue: o Enqueue: The enqueue operation is used to insert the element at the rear

end of the queue. It returns void.

o Dequeue: The dequeue operation performs the deletion from the front-end of

the queue. It also returns the element which has been removed from the front

end. It returns an integer value. The dequeue operation can also be designed to

void.

o Peek: This is the third operation that returns the element, which is pointed by

the front pointer in the queue but does not delete it.

o Queue overflow (isfull): When the Queue is completely full, then it shows the

overflow condition.

o Queue underflow (isempty): When the Queue is empty, i.e., no elements are in

the Queue then it throws the underflow condition.

K. J. Somaiya College of Engineering,
Mumbai (A Constituent College of Somaiya Vidyavihar University) **Department of Computer Engineering**

List 5 Real Life applications of Queue:

• A queue of people at ticket-window: The person who comes first gets the ticket first. The person who is coming last is getting the tickets in last. Therefore, it follows first-in-first-out (FIFO) strategy of queue.

• Vehicles on toll-tax bridge: The vehicle that comes first to the toll tax booth leaves the booth first. The vehicle that comes last leaves last. Therefore, it follows first-in-first-out (FIFO) strategy of queue.

• Phone answering system: The person who calls first gets a response first from the phone answering system. The person who calls last gets the response last. Therefore, it follows first-in-first-out (FIFO) strategy of queue. • Luggage checking machine: Luggage checking machine checks the luggage first that comes first. Therefore, it follows FIFO principle of queue. • Patients waiting outside the doctor's clinic: The patient who comes first visits the doctor first, and the patient who comes last visits the doctor last. Therefore, it follows the first-in-first-out (FIFO) strategy of queue.

K. J. Somaiya College of Engineering,
Mumbai (A Constituent College of Somaiya Vidyavihar University) **Department of Computer Engineering**

Define and explain various types of queue with suitable diagram and their application(s):

There are four different types of queues:

- Simple Queue

- Circular Queue

- Priority Queue

- Double Ended Queue

Simple Queue

In a simple queue, insertion takes place at the rear and removal occurs at the front. It strictly follows the FIFO (First in First out) rule.

Circular Queue

In a circular queue, the last element points to the first element making a circular link. The main advantage of a circular queue over a simple queue is better memory

utilization. If the last position is full and the first position is empty, we can insert an element in the first position. This action is not possible in a simple queue.

Priority Queue

A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue.

Insertion occurs based on the arrival of the values and removal occurs based on priority.

Deque (Double Ended Queue)

In a double ended queue, insertion and removal of elements can be performed from either from the front or rear. Thus, it does not follow the FIFO (First In First Out) rule.

Queue ADT:

/* value definition */

Abstract typedef <front, size> Queue

Condition: size != 0 && front = 0 && rear = -1;

/* operator definition */

Abstract Queue isempty()

Post Condition : true if rear < front;

else return false;

Abstract Queue peek()

Pre Condition : isempty() == false;

Post Condition : return front element;

Abstract Queue enqueue(int x)

Pre Condition: size != N

Post Condition : rear = rear + 1;

Queue[rear] = x;

Abstract Queue dequeue(int x)

Pre Condition : isempty() == false;

Post Condition : y = Queue[front];

rear = rear – 1;

front = front + 1;

return y;

Algorithm for Queue operations using array/Linked list : (Write only the algorithm for assigned type)
Application assigned: Book Wishlist

Implement a priority queue to store the priority of the queues and delete a queue by priority.

Type of queue: Priority Queue

Implementation Details:

1) Mention the application assigned to you and explain how you implemented the solution using the assigned type of Queue.

Steps:
1. The execution starts in the main() function where the user is provided with 3 options – Insert element by priority,delete element, display priority queue
Exit

a. If the user chooses Insertion, they are asked to enter the priority. The book is added to the queue on the basis of priority.

b. If the user chooses Deletion, the booking with highest
priority is removed.
c. If the user chooses Display, the priorities of the books are displayed.
d. If the user chooses the Exit option, they can exit from the program.

Program source code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 4

void create_queue();
void insert_element(int);
void delete_element();
void display_priorityqueue();
void check_priority(int);


int pqueue[MAX];
int front, rear;

int main(){

 create_queue();

 printf("----Menu----");
 printf("\n1.Insert book by priority");
 printf("\n2.Delete book");
 printf("\n3.Display book by priority");
```

```
 printf("\nPress any other number to exit");
 do{
 int n, choice;
 printf("\nEnter your choice: ");
 scanf("%d",&choice);

 if(choice == 1){
 printf("\nEnter priority of the book to insert: ");
scanf("%d",&n);
 insert_element(n);
 }
 else if(choice == 2){
 delete_element();
 }
 else if(choice == 3){
 display_priorityqueue();
 }
 else{
 break;
 }

 }
 while(1);
}
```



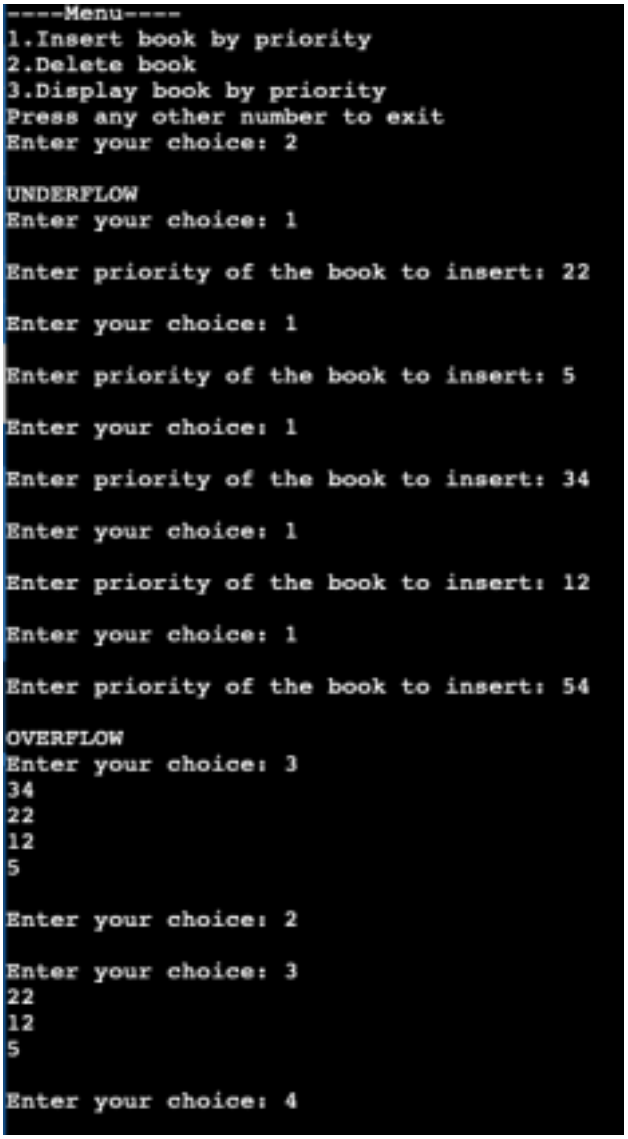
```
void create_queue()
{
 front = rear = -1;
}
void insert_element(int data){
 if (rear >= MAX -1)
 {
 printf("\nOVERFLOW");
 return;
 }
 if ((front == -1) && (rear == -1)){
front++;
 rear++;
 pqueue[rear] = data;
 return;
 }
 else
 check_priority(data);
 rear++;
}
void check_priority(int data)
```

```
{
 int i,j;
 for (i = 0; i <= rear; i++)
 {
 if (data >= pqueue[i])
 {
 for (j = rear + 1; j > i; j--)   {
 pqueue[j] = pqueue[j - 1];   }
 pqueue[i] = data;
 return;
 }
 }
 pqueue[i] = data;
}
void delete_element()
{
 int i;
 if ((front == -1) && (rear == -1))  {
 printf("\nUNDERFLOW");
 return;
 }

 front++;
```



```
 if(front == rear)
 {
 front = rear = -1;
 }
 return;

}
void display_priorityqueue(){
if((front == -1) && (rear == -1)){
printf("\nEmpty queue");  return;
 }
 for(;front <= rear; front++)  {
 printf("%d\n",pqueue[front]);  }
 front = 0;
}
```

Output Screenshots:



Applications of Queue in computer science:

1) When a resource is shared among multiple consumers. Nowadays computer handles multiuser, multiprogramming environment and time-sharing environment. In this environment a system(computer) handles several jobs at a time, to handle these jobs the concept of a queue is used.

2) When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc. 3) In Operating systems:

a) FCFS (first come first serve) scheduling, example:

FIFO queue

b) Spooling in printers 4) In Networks:

a) Queues in routers/ switches

b) Mail Queues

Conclusion:-

In this experiment, we learned about the implementation of queues.