Batch: A1	**Roll No.: 16010123012**
Experiment / assignment / tutorial No. 5	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of Queue operations (Static and Dynamic implementation)- Queue, circular queue, priority queue, and deque

Objective: To implement Basic Operations of Queues

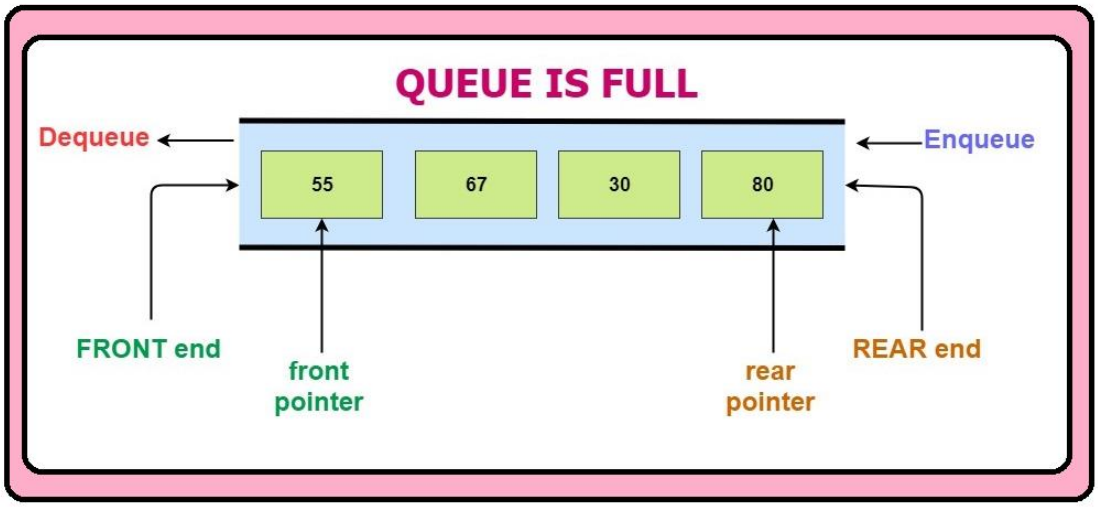
Expected Outcome of Experiment:

CO	Outcome
2	Apply linear and non-linear data structure in application development.

Books/ Journals/ Websites referred:

Introduction:

(diagram of queue)



Program source code:
1) Queue Using Array

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int queue[MAX];
int front = -1, rear = -1;
void insert() {
   if (rear == MAX - 1) {
      printf("Queue is full\n");
      return;
   }
   int data;
   printf("Enter data: ");
   scanf("%d", &data);
   if (front == -1) {
      front = 0;
         rear =0;
   }
   queue[rear++] = data;
   printf("Element inserted successfully.\n");
}
void delete() {
   if (front == -1 || front > rear) {
      printf("Queue is empty!\n");
      return;
   }
```

```
    printf("Deleted element: %d\n", queue[front]);
    front++;
    if (front > rear) {
        front = rear = -1;
    }
}
void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Queue elements:\n");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}
int main() {
    int choice;
    printf("MENU\n");
    while (1) {
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                insert();
```

```
            break;
        case 2:
            delete();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting...\n");
            exit(0);
        default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

2) Queue using Linked list

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node {
   int data;
   struct node *next;
}node;
node *front = NULL;
node *rear = NULL;
void insert() {
   node *ptr;
```

```
    int item;
    ptr = (node *)malloc(sizeof(node));
    if (ptr == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    printf("Enter value: ");
    scanf("%d", &item);
    ptr->data = item;
    ptr->next = NULL;
    if (front == NULL) {
        front = ptr;
        rear = ptr;
    } else {
        rear->next = ptr;
        rear = ptr;
    }
    printf("Element inserted.\n");
}
void delete() {
    node *ptr;
    if (front == NULL) {
        printf("\nQueue Underflow.\n");
        return;
    } else {
        ptr = front;
        front = front->next;
```

```
        free(ptr);
    }
    if (front == NULL) {
        rear = NULL;
    }
    printf("Element deleted.\n");
}
void display() {
    node *ptr;
    if (front == NULL) {
        printf("\nQueue is empty.\n");
    } else {
        printf("Queue contents:\n");
        ptr = front;
        while (ptr != NULL) {
            printf("%d ", ptr->data);
            ptr = ptr->next;
        }
    }
    printf("\n");
}
void exitProgram() {
    node *temp;
    while (front != NULL) {
        temp = front;
        front = front->next;
        free(temp);
```

```
    }
    rear = NULL;
    printf("\nExited program");
    exit(0);
}
int main() {
    int choice = 0;
    printf("MENU");
    while (1) {
        printf("\n1. Insert an element\n2. Delete an element\n3. Display the queue\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
            case 4:
                exitProgram();
                break;
            default:
                printf("\nInvalid choice\n");
```

```
            break;
        }
    }
    return 0;
}
```

3) **Circular Queue**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int CQ[MAX];
int front = -1, rear = -1;
void insert() {
    int val;
    if ((front == 0 && rear == MAX - 1) || (rear == (front - 1) % (MAX - 1))) {
        printf("Queue is full\n");
        return;
    } else {
        printf("Enter the value to insert: ");
        scanf("%d", &val);
        if (front == -1) {
            front = rear = 0;
        } else if (rear == MAX - 1 && front != 0) {
            rear = 0;
        } else {
            rear++;
        }
        CQ[rear] = val;
```

```
        printf("Inserted : %d\n", val);
    }
}
void delete() {
    if (front == -1) {
        printf("Queue is empty\n");
        return;
    }
    int val = CQ[front];
    if (front == rear) {
        front = rear = -1;
    } else if (front == MAX - 1) {
        front = 0;
    } else {
        front++;
    }
    printf("Deleted : %d\n", val);
}
void display() {
    if (front == -1) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements are:\n");
    int i;
    if (rear >= front) {
```

```
        for (i = front; i <= rear; i++) {
            printf("%d ", CQ[i]);
        }
    } else {
        for (i = front; i < MAX; i++) {
            printf("%d ", CQ[i]);
        }
        for (i = 0; i <= rear; i++) {
            printf("%d ", CQ[i]);
        }
    }
    printf("\n");
}
void EXIT(){
printf("Program exited.\n");
exit(0);
}
int main() {
    int choice;
    printf("MENU");
    while (1) {
        printf("\n1. Insert an element\n2. Delete an element\n3. Display the queue\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                insert();
```

```
            break;
        case 2:
            delete();
            break;
        case 3:
            display();
            break;
        case 4:
            EXIT();
            break;
        default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

4) **Doubly – ended Queue**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int deque[MAX];
int front = -1, rear = -1;
void insert() {
   int val, choice;
   if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
      printf("Deque is full\n");
      return;
```

```
}
printf("Enter the value to insert: ");
scanf("%d", &val);
printf("Where do you want to insert the value?\n1. Front\n2. Rear\n");
printf("Enter your choice: ");
scanf("%d", &choice);
if (choice == 1) {
    if (front == -1) {
        front = rear = 0;
    } else if (front == 0) {
        front = MAX - 1;
    } else {
        front--;
    }
    deque[front] = val;
    printf("Inserted %d at the front\n", val);
} else if (choice == 2) {
    if (rear == -1) {
        front = rear = 0;
    } else if (rear == MAX - 1) {
        rear = 0;
    } else {
        rear++;
    }
    deque[rear] = val;
    printf("Inserted %d at the rear\n", val);
} else {
```

```
        printf("Invalid choice!\n");
    }
}
void delete() {
    int choice;
    if (front == -1) {
        printf("Deque is empty!\n");
        return;
    }
    printf("From where do you want to delete the value?\n1. Front\n2. Rear\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    if (choice == 1) {
        int val = deque[front];
        if (front == rear) {
            front = rear = -1;
        } else if (front == MAX - 1) {
            front = 0;
        } else {
            front++;
        }
        printf("Deleted %d from the front\n", val);
    } else if (choice == 2) {
        int val = deque[rear];
        if (front == rear) {
            front = rear = -1;
        } else if (rear == 0) {
```

```
            rear = MAX - 1;
        } else {
            rear--;
        }
        printf("Deleted %d from the rear\n", val);
    } else {
        printf("Invalid choice!\n");
    }
}
void display() {
    if (front == -1) {
        printf("Deque is empty!\n");
        return;
    }
    printf("Deque elements are:\n");
    int i = front;
    if (front <= rear) {
        while (i <= rear) {
            printf("%d ", deque[i++]);
        }
    } else {
        while (i <= MAX - 1) {
            printf("%d ", deque[i++]);
        }
        i = 0;
        while (i <= rear) {
            printf("%d ", deque[i++]);
```

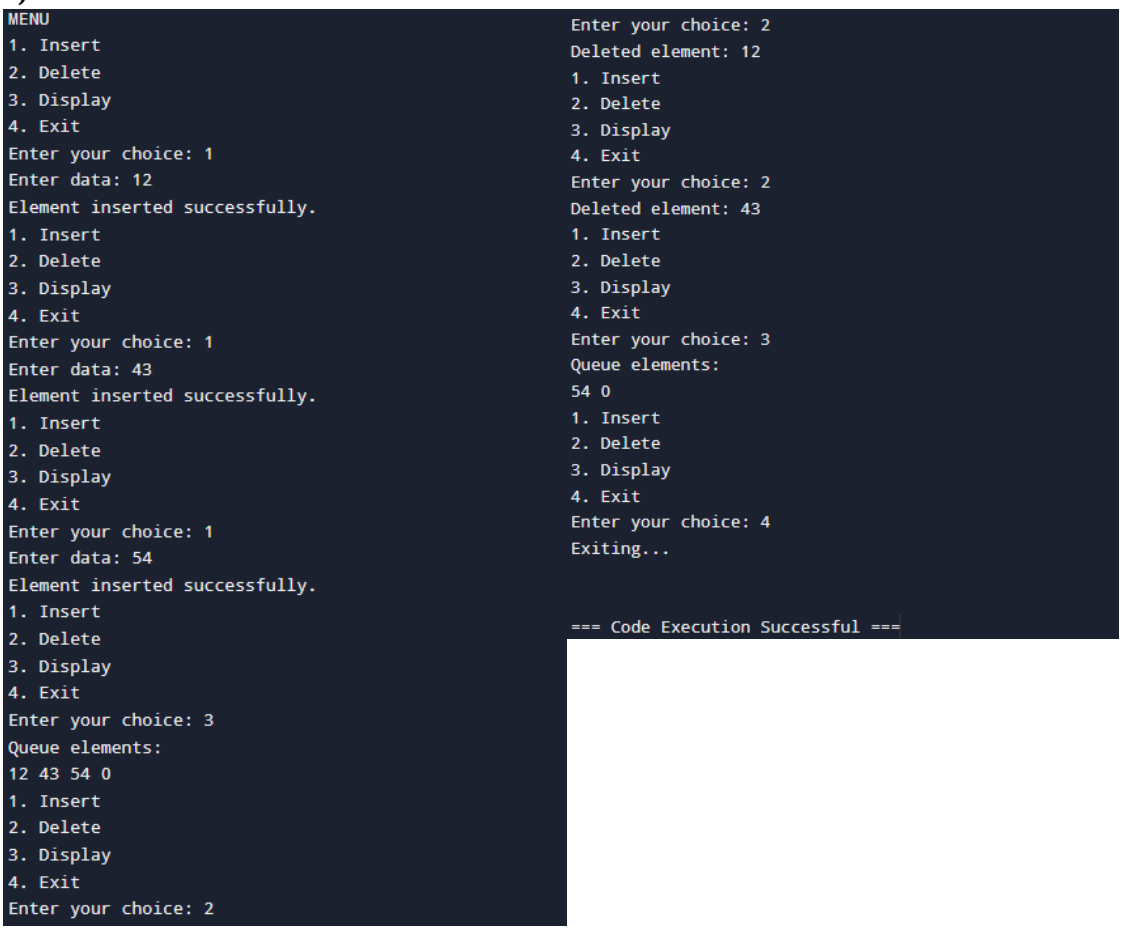
```
        }
    }
    printf("\n");
}
int main() {
    int choice;
    while (1) {
        printf("\n1. Insert an element\n2. Delete an element\n3. Display the queue\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exited program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
```

```
    return 0;

}
```

Output Screenshots:

1)



2)

```
MENU
1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter value: 12
Element inserted.

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter value: 23
Element inserted.

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Queue contents:
12 23

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 2
Element deleted.

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Queue contents:
23

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 4

Exited program

=== Code Execution Successful ===
```

3)

```
MENU
1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 342
Inserted : 342

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 45
Inserted : 45

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 67
Inserted : 67

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Queue elements are:
342 45 67

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 2
Deleted : 342

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Queue elements are:
45 67

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 4
Program exited.
```

4)

```
1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 54
Where do you want to insert the value?
1. Front
2. Rear
Enter your choice: 1
Inserted 54 at the front

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 2
Where do you want to insert the value?
1. Front
2. Rear
Enter your choice: 2
Inserted 2 at the rear

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Deque elements are:
54 2

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter the value to insert: 43
Where do you want to insert the value?
Where do you want to insert the value?
1. Front
2. Rear
Enter your choice: 1
Inserted 43 at the front

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 2
From where do you want to delete the value?
1. Front
2. Rear
Enter your choice: 2
Deleted 2 from the rear

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Deque elements are:
43 54

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 4
Exited program


=== Code Execution Successful ===
```

Conclusion:

We learned to implement different types od Queues.

Post lab questions:

1. What are the key considerations in choosing between a circular array and a linked list for the implementation of a queue?
   Key considerations when choosing a circular array include:
   Fixed size with faster access due to contiguous memory, making it cache-friendly.
   Limited by predefined capacity, making it efficient for small or fixed-size queues.
   Simple to implement but requires careful handling of wrap-around logic.
   Key considerations when choosing a linked list include:
   Dynamic size without a fixed limit, offering flexibility.
   Slower due to pointer traversal and additional memory overhead for storing pointers.
   Suitable for queues with unpredictable or large size variations, though implementation is more complex.

2. Implementation of Priority Queue.

```
#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int queue[MAX];

int priority[MAX];

int size = 0;

void insert() {

   if (size == MAX) {

      printf("Queue is full\n");

      return;

   }

   int data, pr;

   printf("Enter data: ");

   scanf("%d", &data);

   printf("Enter priority: ");

   scanf("%d", &pr);
```

```
    int i = size - 1;
    while (i >= 0 && priority[i] < pr) {
        queue[i + 1] = queue[i];
        priority[i + 1] = priority[i];
        i--;
    }
    queue[i + 1] = data;
    priority[i + 1] = pr;
    size++;
    printf("Element inserted successfully.\n");
}
void deleteMax() {
    if (size == 0) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Deleted element with data: %d, priority: %d\n", queue[0], priority[0]);
    int i;
    for (i = 1; i < size; i++) {
        queue[i - 1] = queue[i];
        priority[i - 1] = priority[i];
    }
    size--;
}
void display() {
    if (size == 0) {
```

```
        printf("Queue is empty!\n");
        return;
    }
    printf("Priority Queue:\n");
    int i;
    for (i = 0; i < size; i++) {
        printf("Data: %d, Priority: %d\n", queue[i], priority[i]);
    }
}
int main() {
    int choice;
    printf("MENU");
    while (1) {
        printf("\n1. Insert an element\n2. Delete an element\n3. Display the queue\n4.
Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                insert();
                break;
            case 2:
                deleteMax();
                break;
            case 3:
                display();
```

```
            break;
        case 4:
            printf("Exited program\n");
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
      }
   }
   return 0;
}
```

```
MENU
1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter data: 21
Enter priority: 2
Element inserted successfully.

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 1
Enter data: 32
Enter priority: 1
Element inserted successfully.

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 3
Priority Queue:
Data: 21, Priority: 2
Data: 32, Priority: 1

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 2
Deleted element with data: 21, priority: 2

1. Insert an element
2. Delete an element
3. Display the queue
4. Exit
Enter your choice: 4
Exited program
```