**Aim ↬ To perform basic operations on Queue.**

**Objectives ↬**

i. Write a Program in C to implement queue and its operations (insertion, deletion and traverse) using arrays.
ii. Write a program to implement the following types of DEQUEUE using Array Implementation:
    a. Input Restricted Dequeue.
    b. Output Restricted Dequeue.
    c. Unrestricted Dequeue.

**Software Required ↬** Visual Studio Code

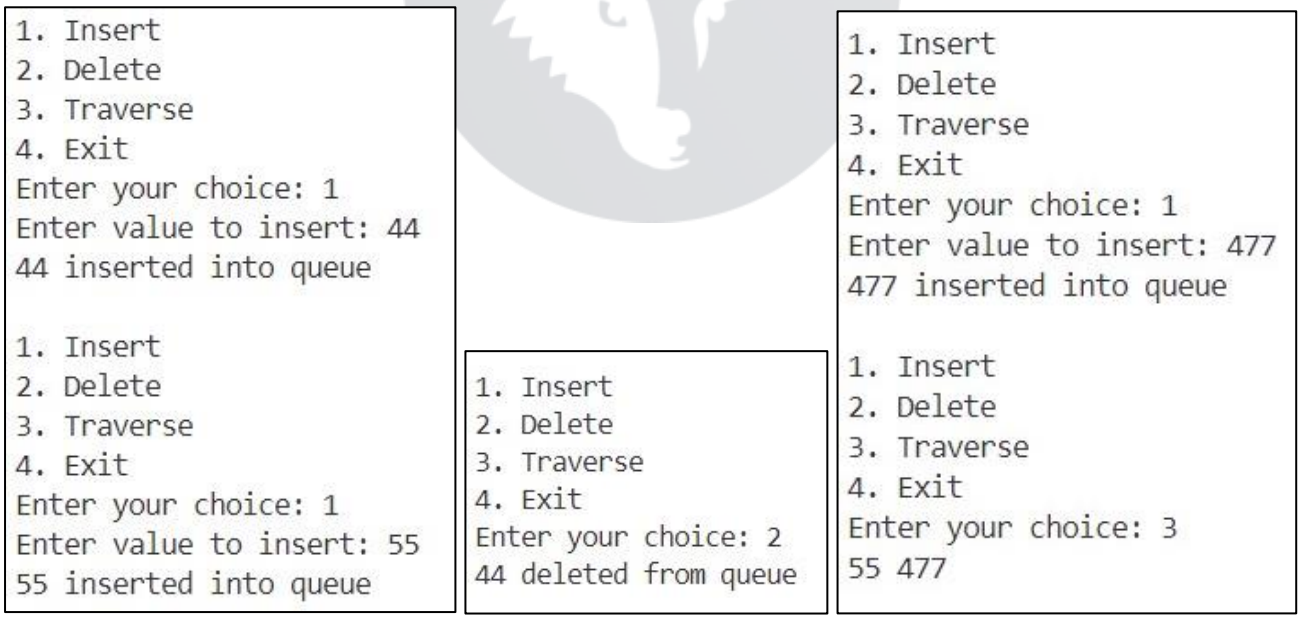
**Code 1 ↬**

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 5

int queue[MAX];
int front = -1, rear = -1;

int isFull() {
   return (rear == MAX - 1);
}

int isEmpty() {
   return (front == -1 || front > rear);
}

void insert(int value) {
   if (isFull()) {
      printf("Queue Overflow\n");
      return;
   }
   if (front == -1) {
      front = 0;
   }
```

```
    queue[++rear] = value;
    printf("%d inserted into queue\n", value);
}

int delete() {
    if (isEmpty()) {
        printf("Queue Underflow\n");
        return -1;
    }
    return queue[front++];
}

void traverse() {
    if (isEmpty()) {
        printf("Queue is empty\n");
        return;
    }
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int choice, value;

    while (1) {
        printf("\n1. Insert\n2. Delete\n3. Traverse\n4. Exit\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insert(value);
                break;
            case 2:
```

```
            value = delete();
            if (value != -1) {
                printf("%d deleted from queue\n", value);
            }
            break;
        case 3:
            traverse();
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice\n");
      }
   }

   return 0;
}
```

## Output ↬



## Code 2 ↬

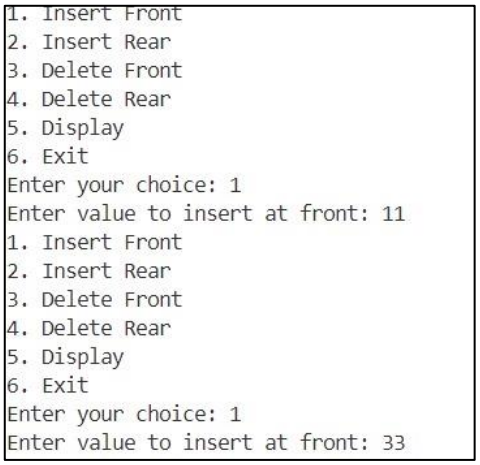
```
#include <stdio.h>
#define MAX 100

typedef struct {
    int arr[MAX];
    int front, rear;
} Dequeue;

void init(Dequeue *dq) {
    dq->front = dq->rear = -1;
}

int isFull(Dequeue *dq) {
    return (dq->front == 0 && dq->rear == MAX - 1) || (dq->front == dq->rear + 1);
}

int isEmpty(Dequeue *dq) {
    return dq->front == -1;
}

void insertFront(Dequeue *dq, int value) {
    if (isFull(dq)) {
        printf("Dequeue is full\n");
        return;
    }
    if (dq->front == -1) {
        dq->front = dq->rear = 0;
    } else if (dq->front == 0) {
        dq->front = MAX - 1;
    } else {
        dq->front--;
    }
    dq->arr[dq->front] = value;
}

void insertRear(Dequeue *dq, int value) {
```

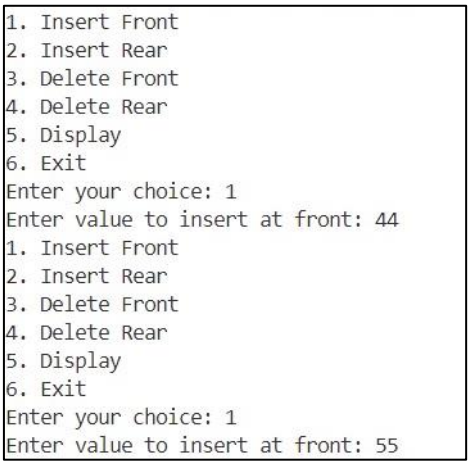
```
    if (isFull(dq)) {
        printf("Dequeue is full\n");
        return;
    }
    if (dq->front == -1) {
        dq->front = dq->rear = 0;
    } else if (dq->rear == MAX - 1) {
        dq->rear = 0;
    } else {
        dq->rear++;
    }
    dq->arr[dq->rear] = value;
}

void deleteFront(Dequeue *dq) {
    if (isEmpty(dq)) {
        printf("Dequeue is empty\n");
        return;
    }
    if (dq->front == dq->rear) {
        dq->front = dq->rear = -1;
    } else if (dq->front == MAX - 1) {
        dq->front = 0;
    } else {
        dq->front++;
    }
}

void deleteRear(Dequeue *dq) {
    if (isEmpty(dq)) {
        printf("Dequeue is empty\n");
        return;
    }
    if (dq->front == dq->rear) {
        dq->front = dq->rear = -1;
    } else if (dq->rear == 0) {
        dq->rear = MAX - 1;
```

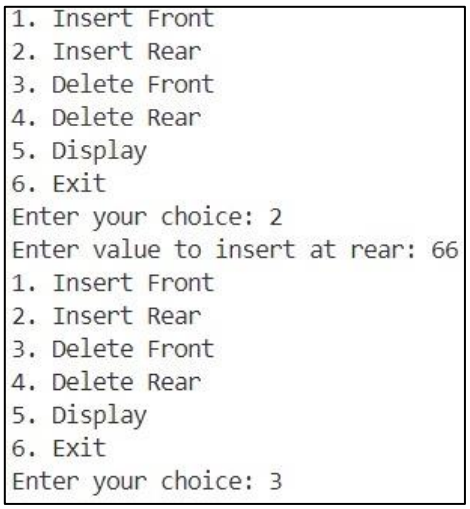
```
    } else {
        dq->rear--;
    }
}

void display(Dequeue *dq) {
    if (isEmpty(dq)) {
        printf("Dequeue is empty\n");
        return;
    }
    int i = dq->front;
    while (1) {
        printf("%d ", dq->arr[i]);
        if (i == dq->rear) break;
        i = (i + 1) % MAX;
    }
    printf("\n");
}

int main() {
    Dequeue dq;
    init(&dq);

    int choice, value;

    while (1) {
        printf("1. Insert Front\n");
        printf("2. Insert Rear\n");
        printf("3. Delete Front\n");
        printf("4. Delete Rear\n");
        printf("5. Display\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

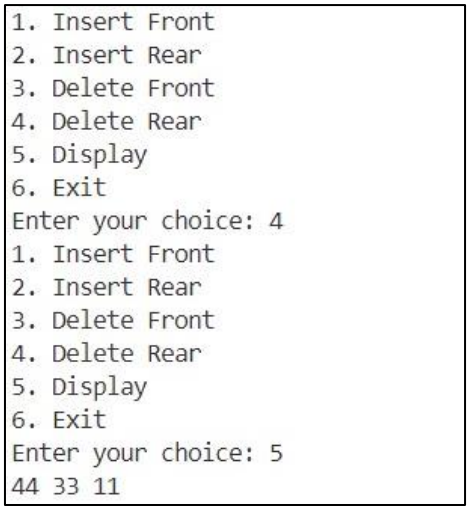
```
            printf("Enter value to insert at front: ");
            scanf("%d", &value);
            insertFront(&dq, value);
            break;
        case 2:
            printf("Enter value to insert at rear: ");
            scanf("%d", &value);
            insertRear(&dq, value);
            break;
        case 3:
            deleteFront(&dq);
            break;
        case 4:
            deleteRear(&dq);
            break;
        case 5:
            display(&dq);
            break;
        case 6:
            return 0;
        default:
            printf("Invalid choice\n");
      }
   }
   return 0;
}
```

## Output ↬






## Result ↬

The programs demonstrated:

- **Queue Operations**: Successful implementation of insertion, deletion, and traversal using arrays.
- **Deque Operations**: Correct implementation of Input Restricted, Output Restricted, and Unrestricted Dequeues using arrays.

## Conclusion ↬

The experiment effectively illustrated the implementation of basic queue and deque operations using arrays, providing insights into dynamic and static data structure management.

## Precautions ↬

- Validate inputs and manage memory effectively.
- Handle edge cases like empty queues or deques.
- Implement proper error handling for overflow and underflow conditions.
- Implement error handling for missing nodes or keys.