3a WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display The program should print appropriate messages for queue empty and queue overflow conditions

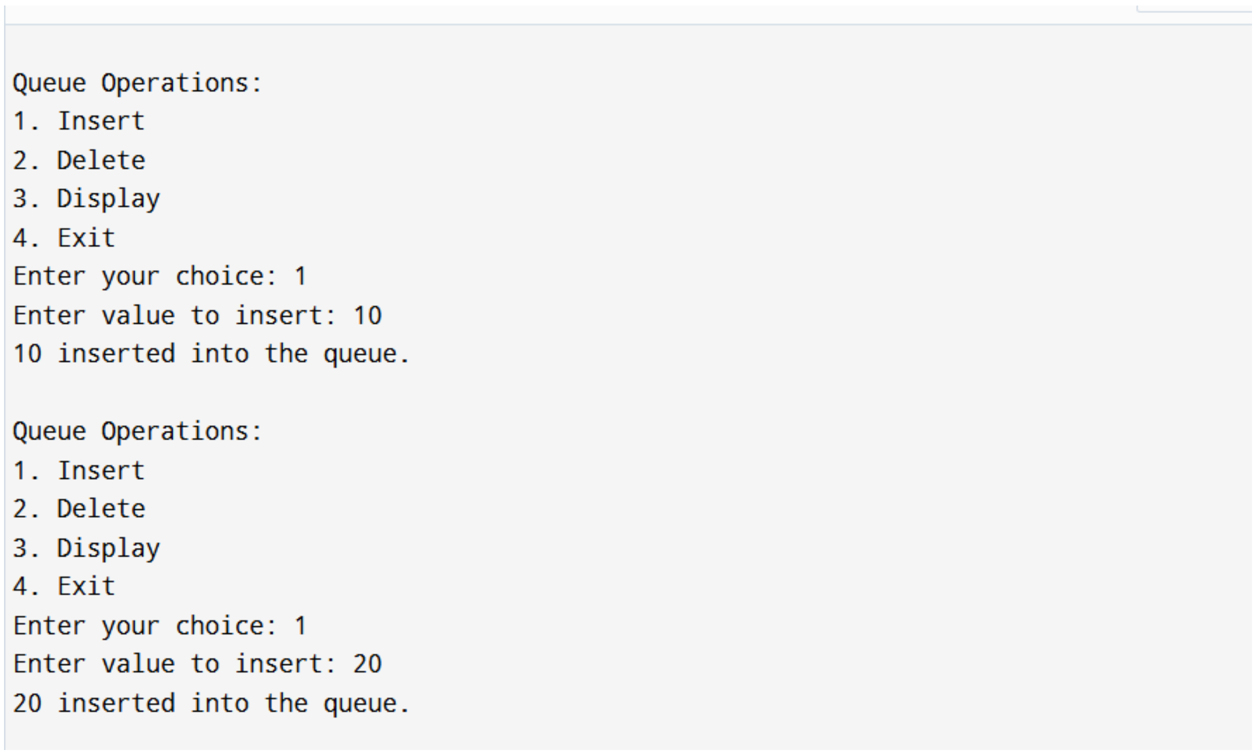
```
#include <stdio.h>

#define MAX 5   // maximum size of the queue


int queue[MAX];

int front = -1, rear = -1;


void insert(int value)

{

  if (rear == MAX - 1)

  {

    printf("Queue Overflow! Cannot insert %d\n", value);

  }

  else

  {

    if (front == -1)

    {

      front = 0;

    }

    rear++;

    queue[rear] = value;

    printf("%d inserted into the queue.\n", value);

  }

}
```

```
void delete()
{
    if (front == -1 || front > rear)
    {
        printf("Queue Underflow! Queue is empty.\n");
    }
    else
    {
        printf("Deleted element: %d\n", queue[front]);
        front++;
    }
}

void display()
{
    if (front == -1 || front > rear)
    {
        printf("Queue is empty.\n");
    }
    else
    {
        printf("Queue elements: ");
        for (int i = front; i <= rear; i++)
        {
```

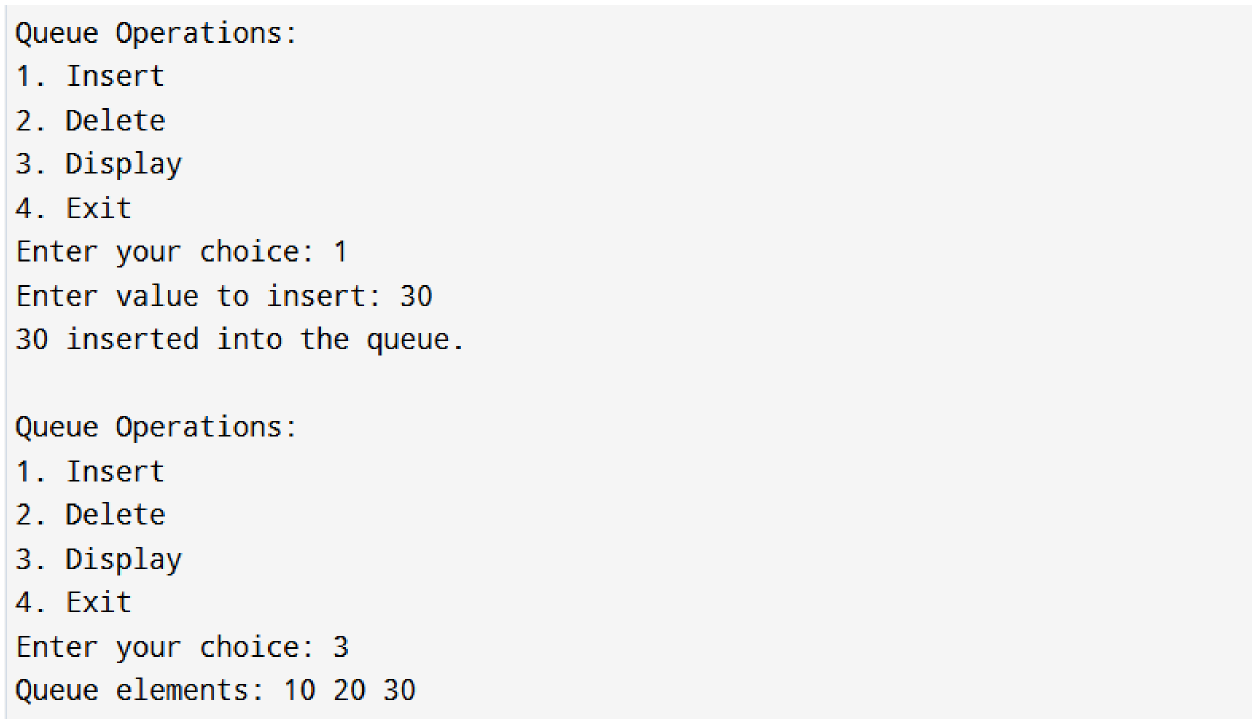
```
      printf("%d ", queue[i]);
    }
    printf("\n");
  }
}

int main()
{
  int choice, value;

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

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

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

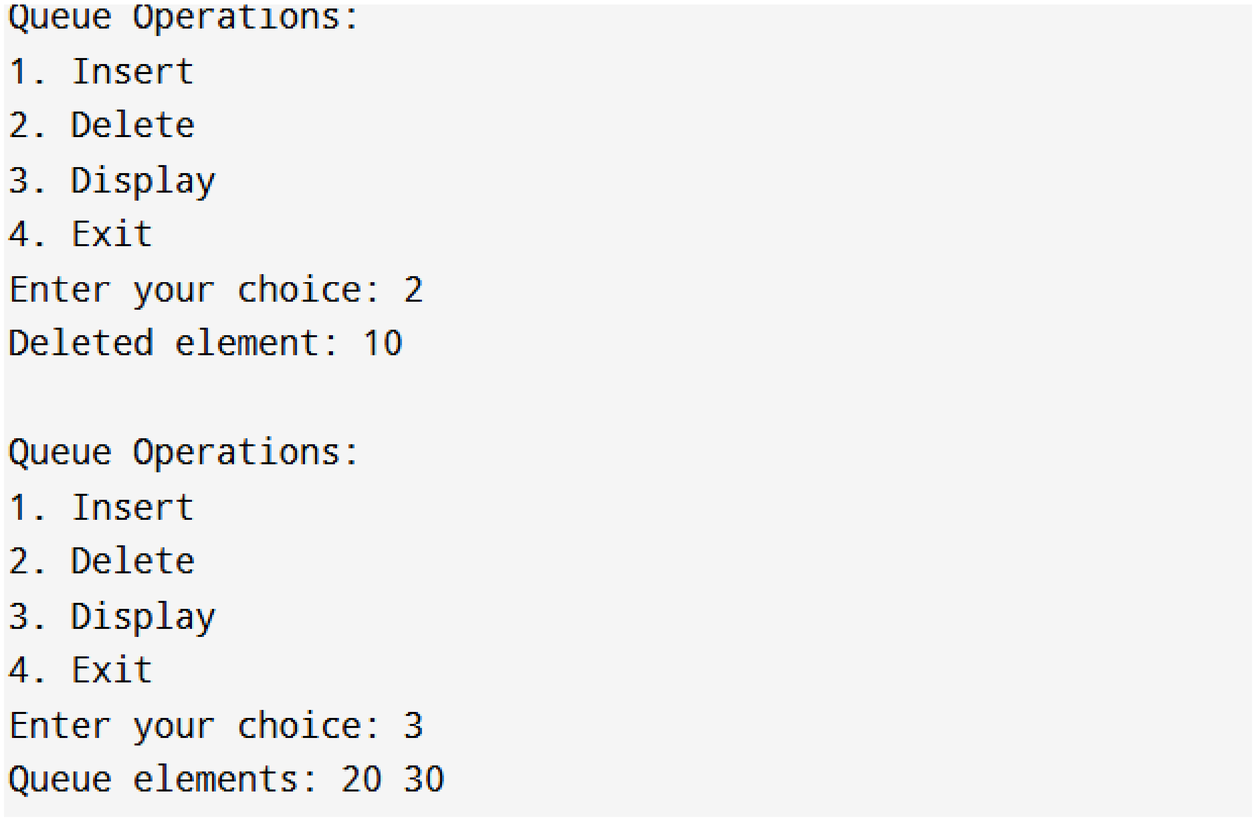
Output



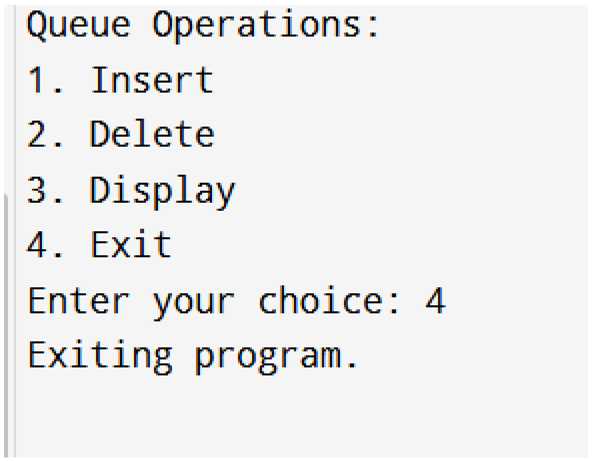
```
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 10
10 inserted into the queue.

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 20
20 inserted into the queue.
```



```
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 30
30 inserted into the queue.

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 10 20 30
```



```
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted element: 10

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 20 30
```



```
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4
Exiting program.
```

3b WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions

```c
#include <stdio.h>

#define MAX 5
```

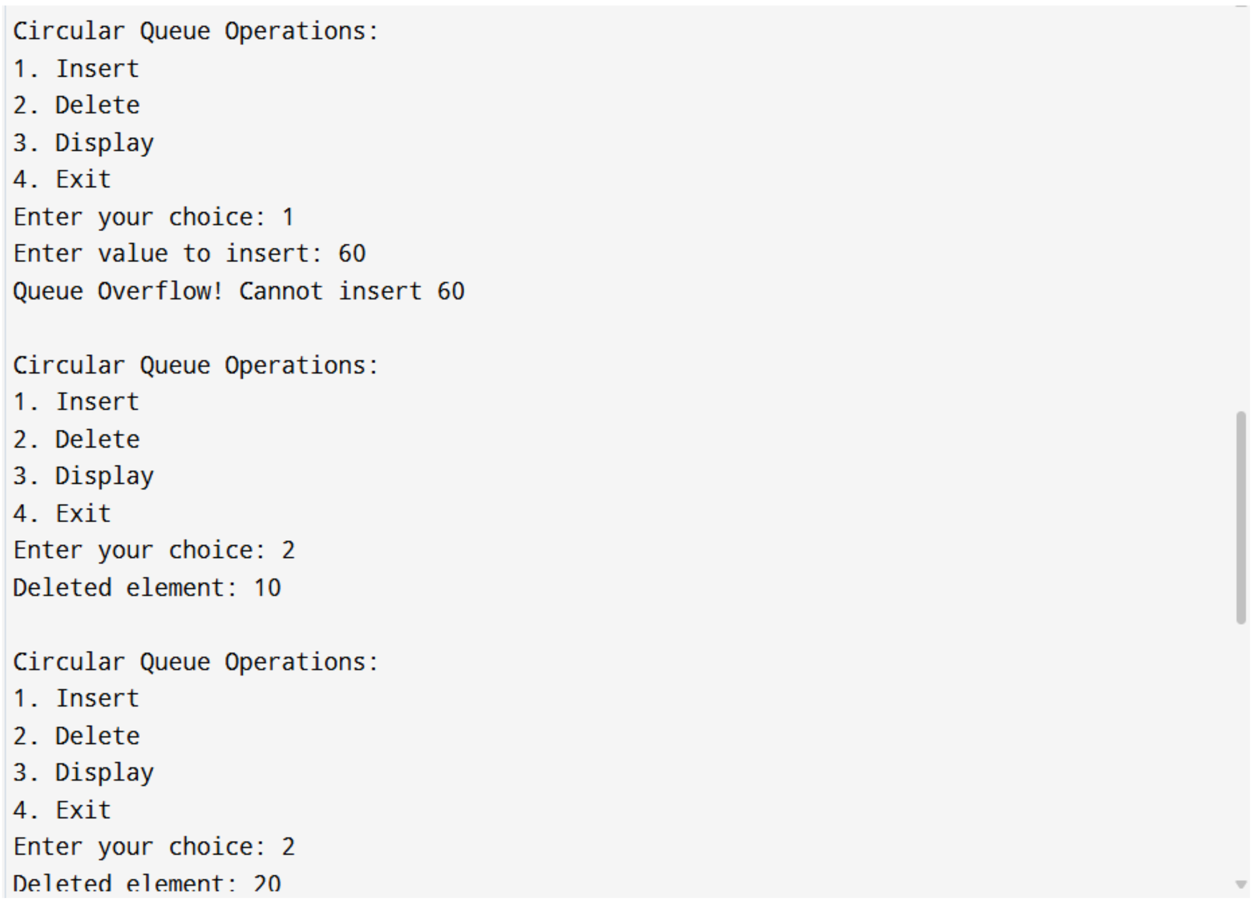
```
int queue[MAX];

int front = -1, rear = -1;

void insert(int value)

{

  if ((front == 0 && rear == MAX - 1) || (front == (rear + 1) % MAX))

  {

    printf("Queue Overflow! Cannot insert %d\n", value);

  }

  else

  {

    if (front == -1)

    {

      front = 0;

      rear = 0;

    }

    else

    {

      rear = (rear + 1) % MAX;

    }

    queue[rear] = value;

    printf("%d inserted into the queue.\n", value);

  }

}

void delete()

{

  if (front == -1)
```

```
    {
        printf("Queue Underflow! Queue is empty.\n");
    }
    else
    {
        printf("Deleted element: %d\n", queue[front]);
        if (front == rear)
        {

            front = -1;
            rear = -1;
        }
        else
        {
            front = (front + 1) % MAX;
        }
    }
}

void display()
{
    if (front == -1)
    {
        printf("Queue is empty.\n");
    }
    else
    {
        printf("Queue elements: ");
        int i = front;
        while (1)
        {
            printf("%d ", queue[i]);
            if (i == rear)
                break;
            i = (i + 1) % MAX;
        }
        printf("\n");
    }
}

int main()
{
    int choice, value;

    while (1)
    {
        printf("\nCircular Queue Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
          case 1:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            insert(value);
            break;
          case 2:
            delete();
            break;
          case 3:
            display();
            break;
          case 4:
            printf("Exiting program.\n");
            return 0;
          default:
            printf("Invalid choice! Please try again.\n");
        }
      }
      return 0;
    }
```

Output

```
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 10
10 inserted into the queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 20
20 inserted into the queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
```

```
Enter your choice: 1
Enter value to insert: 30
30 inserted into the queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 40
40 inserted into the queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 50
50 inserted into the queue.
```



```
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 60
Queue Overflow! Cannot insert 60

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted element: 10

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Deleted element: 20
```

```
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter value to insert: 60
60 inserted into the queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 30 40 50 60

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4
Exiting program.
```