3. WAP to stimulate 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:

Pseudocode:

```
# define N 5
int queue [N]
int front = -1
int rear = -1
void enque (int x)
 {
① if (rear == N-1)
  { printf ("Queue Overflow");
  }
② else if (front == -1 && rear == -1)
      { front = rear = 0;
        queue [rear] = x;
      }
③ else {
      rear ++ ;
      queue [rear] = x;
  }
 }
④ Void dequeue ()
  { if (front == -1 && rear == -1)
        printf ("Queue is Empty");
⑤ else if (front == rear)
    { front = rear = -1;
  }
  else { printf ("Deleted elem %d", queue [front]);
```

```
        front++;
    }
}
```

⑥
```
void display()
{
    int i
    else {
        for (i = front; i < rear ; i++)
        {
        printf ("%d", queue [i]);
    }
```

⑦
```
void peek()
{
    front == -1 &&
    (if . rear == -1)
    {
        printf ("queue is empty");
    }
    else
        {
            printf ("%d", queue [front]);
```

⑧
```
Switch (choice):
    case 1:
        Call enqueue()
        Break
    case 2:
        Call dequeue()
        Breake
    case 3:
        Call peek()
        Break
    case 4:
        Call display()
        Break
    case 5:
        Display "Exiting program...!"
        Terminate program
```

Display "Invalid choice!"

Code:

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

# define SIZE 5
int queue [SIZE];
int front = -1 , rear = -1;
void enqueue () {
 int value;
  if (rear = = SIZE -1 ){
 printf ("\n Queue is FULL ! Cannot insert more elements.\n");
 } else {
       print f ("Enter the value to enqueue: ");
       Scan f ("%.d ", & value);
     if (front = = -1) {
       front = 0;
     }
       rear ++;
       queue [rear ] = value ;
       printf ("%d inserted into the queue.\n", value);
   }
 }
   void dequeue ( ) {
    if (front = = -1 || front > rear ){
    printf ("\n Queue is EMPTY ! Nothing to remove.\n");
    }else {
       printf ("%.d removed from the queue.\n",
                                          Queue [front]
```

```
front ++;
if (front > rear) {
front = rear = -1;
}
}
}
void peek() {
  if (front == -1 || front > rear) {
   printf ("\n Queue is EMPTY!\n");
  } else {
   printf ("\n Front element is: %d \n",
                          queue [front]);
  }
}
void display () {
if (front == -1 || front > rear) {
 printf ("\n Queue is EMPTY!\n");
} else {
     printf("\n Queue elements are: ");
  for (int i = front; i <= rear; i++) {
    printf (" %d", queue [i]);
  }
  printf("\n");
 }
}
  int main () {
  int choice;
  while (1) {
```

```
printf("1. Enqueue (INSERT)\n");
printf("2. Dequeue (Remove)\n");
printf("3. Peek (Front Element)\n");
printf("4. Display Queue\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
    case 1:
        enqueue();
        break;
    case 2:
        dequeue();
        break;
    case 3:
        peek();
        break;
    case 4:
        display();
        break;
    case 5:
        printf("\nExiting Program... !Good bye!\n");
        exit(0);
    default:
        printf("\nInvalid choice! Please try again.\n");
}
}
return 0;
}
```

Output :

```
QUEUE OPERATIONS MENU
1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front element)
4. Display Queue
5. Exit
Enter your choice : 3
Queue is EMPTY ! ~~Nothing to peek~~.
QUEUE OPERATIONS MENU
1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit
Enter your choice : 1
Enter the value to enqueue : 1
1 is inserted into the queue.

QUEUE OPERATIONS MENU
1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit
Enter your choice : 1
Enter the value to enqueue : 2
2 is inserted into the queue.
```

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit

Enter your choice: 1
Enter the value to enqueue: 3
3 is inserted into the queue.

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit

Enter your choice: 1
Enter the value to enqueue: 4
4 is inserted into the queue.

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit

Enter your choice: 1
Enter the value to enqueue: 5
5 inserted into the queue.

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue
5. Exit

Enter your choice: 1

Queue is FULL! Cannot insert more elements.

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue

Enter your choice: 2

1 removed from the Queue.

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove)
3. Peek (Front Element)
4. Display Queue

MG 13/10/25

Enter your choice: 4

Queue elements are: 2 3 45

QUEUE OPERATIONS MENU

1. Enqueue (Insert)
2. Dequeue (Remove
3. Peek (Front Element)
4. Display Queue
5. Exit

Enter your choice: 5