```
Enter your choice : 3
Queue items are : 10, 11, 12, 13, 14
- - - - - - - - - - - -
Enter your choice : 2
value of 10 deleted from the queue.
- - - - - - - - - -
Enter your choice : 3
Queue items are : 11, 12, 13, 14.
- - - - - - - - - - - -
Enter your choice : 4
  exiting program.
```

M4
3/11/25.

```
output:

option Available:
1. Enqueue
2  Dequeue
3  Display
4  Exit

Enter your choice : 2
queue is empty, underflow
----- -----

Entere your choice : 1
Enter value to insert : 10
value of 10 inserted in the queue.
- - - - - - - -

Entere your choice : 1.
Enter value to insert : 11
value of 11 inserted in the queue.
- - - - - - - - -

Enter your choice : 1
Enter value to inseret : 12
value of 12 inseeted in the queue.
- - - - - - - - - - - - - -

Enter your choice : 2
Enter value to insert : 13
value of 13 inserted in the queue.
- - - - - - - - - - - -

Enter your choice : 1.
Enter value to insert : 14
value of 14 inserted in the queue.
- - - - - - - - - - - -

Entere your choice : 1
Entere value to insert : 15
Queue overflow
- - - - - - - - -
```

```
printf ("1. Enqueue \n 2. Dequeue \n
        3. Display \n 4. Exit \n");

while (1) {
    printf ("\n ---- --- \n");
    printf (" Enter your choice :");
    scanf (" %d ", & choice );

    switch (choice ) {

     case 1 :
          printf (" Enter value to insert :");
          scanf (" %d ", & value );
          enqueue (value );
           break;

     case 2 :
               dequeue ( );
                break;

     case 3 ;
               display ( );
                break;

     case 4 ;
              printf (" Exiting program .\n");
              return 0;

     default :
                  printf (" Invalid choice");
     }
   }
}
```

```
int item = queue[front];
if (front == rear)
{
    front = rear = -1;
}
else {
        front = (front +1) % size;
}.
 printf("\n value of %d deleted
              from the queue.", item);
  return item;
 }.

void display()
{
 if ( front == -1)
 {
   printf("\n queue is empty.");
   return;
 }.
   printf("\n queue items are :");
   int i = front;
   while (i != rear)
    {
       printf("%d ", queue[i]);
        i = (i+1) % size;
       }
       printf("%d\n", queue[rear]);
    }.

int main()
{
  int choice, value;
  printf("\n options Available:\n");
```

```
# include <stdio.h>
# define size 5

int queue [size];
int front = -1;
int rear = -1;

void enqueue (int value)
{
  if ( (rear + 1) % size == front )
  {
   printf (" \n queue overflow ");
    return;
  }.
   if (front == -1) {
      front = rear = 0;
   }
   else {
        rear = (rear + 1) % size;
    }.
    queue [rear] = value;
    printf (" \n value of %d inserted in
              the queue.", value);
  }.

int dequeue ()
{
  if (front == -1) {
      printf (" \n queue is empty, under
                         flow");
      return -1;
  }.
```

3b

Q 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 message for queue empty, queue overflow.

```
# pseudo code
# define N 5
int queue[N]
int front = -1
int rear = -1
void enqueue (int x)
{
    if (~~rear == N-1~~) (rear+1)%N == front)
      printf ("Queue overflow");
      }
  else if (rear == -1 && front == -1)
        Set rear = front = 0
            queue(rear) = x; }
    else
            ~~rear++~~ rear = (rear+1)%N
            ~~queue~~ queue[rear] = x;


void ~~deletete~~ display()
 if (rear == -1 && front == -1)
   printf(queue empty)
else if (rear = front)
   Set rear = front = -1.
else
   set i = front.   while (i != rear)
   for (i = front ; i <= rear ; i++)
   {
printf ("%d", queue[~~pos~~ i ].
Set i = (i+1) %N
 printf (" %d", queue [rear]];


void dequeue() {
if (rear == -1 && front == -1)
  printf ("Queue empty");
 else if (rear = front)
   printf (" deleted item is %d", queue[front]);
        rear = front = -1;
 else
      front = (front +1) % N
```

MG