Assignment-6

1. WAP implementing insert, delete and display operations of circular queue.

Soln →

PROGRAM-

Insert function

```
void insert (int item)
{
if ( front ==0 && rear == MAX-1) || (front == rear +1))
 {
 printf ("Queue Overflow \n");
 return;
 }
 if (front ==-1)
 {
      front = 0;
       rear = 0;
  }
  else
  {
      if (rear ==MAX-1)
      rear = 0;
      else
         rear = rear+1;
  }
  cqueue_arr [rear] = item;
}
```

Delete Function-

```
void deletion()
{
  if (front ==-1)
  {
     printf("Queue Underflow");
     return;
  }
  printf("element deleted from queue is:
            %dn", cqueue_arr[front]);
   if (front == rear)
   {
      front =-1;
      rear =-1;

   }
   else
   {
    if (front == MAX-1)
    front =0;
    else
       front = front +1;

    }
}
```

Display Function

PROGRAM-

```
void display()
{
  int front_pos = front, rear_pos = rear;
    if (front ==-1)
    {
      printf("Queue is empty\n");
      return;
    }
   printf("Queue element : n");
   if (front_pos <= rear_pos)
   while (front_pos <= realpos)
    {
      printf("%d", cqueue_arr[front_pos]);
      front_pos++;
    }
    else
    {
      while(front-pos<= MAX-1)
     {
        printf("%d ", cqueue_arr[front_pos]);
        front pos ++;
     }
     front_pos =0;
      while (front_pos <= rear_pos)
      {
        printf("%d", cqueue_arr [front_pos]);
        front pos++;
      }
  }
```

```
        printf("n");
    }
```

Q→

Sol^n - PROGRAM-

```
Struct MyStack
{
    stack <int> s;
    int minEle;

    //prints minimum element of my Stack
    void getMin()
    {
            if (s.empty())
                cout << "stack is empty \n"};
            // variables minEle stores the minimum element
            // in the stack
            else
                cout << "minimum element in the stack is :"
                          << minEle << " \n";
    }
    // prints top element of my stack
    void peek()
    {   if (s.empty())
        {
            cout << "stack is empty ";
            return;
        }
    int t =s.top(); //Topelement.
    cout <<"Top most Element is : ";
    // If t < minEle means min Ele stores
    // value of t.
    (t < minEle)? cout << minEle : cout <<t;
}
// Remove the top element from MY stack
void pop()
 {
    if (s.empty())
    {
        cout <<"stack is empty \n";
        return;
    }
    cout <<"Top most Element Removed:";
    int t = s.top();
    s.pop();

    if (t <minEle)
    {
        cout << minEle <<"\n";
        minEle = 2* minEle -t;
    }
    else
        cout <<t <<"\n";
 }
```

```
// Removes top element from My Stack
  void push (int x)
  {

        // Insert new number into the stack
        if (s.empty())
         {
              min Ele = x;
              s.push (x);
              cout << "Number Inserted: " << x << "\n";
              return;
          }
         // If new number is less than minEle
         if (x < min Ele)
          {
              s.push (2* x - minEle);
               min Ele = x;
           }
           else
            {
             s. push (x);
             }
            cout << "Number Inserted:" << x << "\n";
        }
   };
```

ex-

```
//Driver code
int main()
{
        My stack s;
        s.push(3);
        s.push(5);
        s.get min();
        s.push(2);
        s.push(1);
        s.get Min();
        s.pop();
        s.get min();
        s.pop();
        s.peek();

        return 0;

}
```

Output

```
Number Inserted : 3
Number Inserted : 5
Minimum Element in the stack is : 3
Number Inserted : 2
Number Inserted : 1
Minimum Element in the stack is : 1
Top Most Element Removed : 1
Minimum Element in the stack is : 2
Top Most element Removed : 2
Top most element is : 5
```