DS:

Stack:

```
Void push ( ) {
if (top == Max_stack -1){
        cout << "overflow";
    }
else {
    cout <<
      cin >> x;
      top ++;
      stack[top] = x;
  }
Void POP ( ) {
  if(top == -1)
    {cout << "underflow";
    }
  else {
      cout << stack [top];
      top--;
  }
```

```
void display( ){
    if (top == -1) {
        cout << "stack is empty";
    }
    else {
        for (i=0 ; i <= top ; i++){
            cout << stack [i];
        }
    }
}
```

Queue:

```
void enqueue(int x){
  if (rear = n-1){
    cout<<"overflow";
  }
elseif (front == -1 && rear == -1){
  front = rear = 0;
   queue[rear] = y;
  else {
    rear ++;
    queue [rear] = x;
  }
}
void dequeue (){
 if (front == -1 && rear == -1){
    cout<<" queue is empty";
}else if (front == rear) {
   cout<<" queue[front];
   front = rear = -1;
}
```

```
else {
    cout << queue[front];
    front++;
    }
}

void display(){
    int i;
    if(front == -1 && rear == -1){
        cout << "Queue is empty";
    }
    else{
        for(i=front; i<=rear; i++){
            cout << queue[i];
        }
    }
}
```

Circular Queue:

```
void enqueue(int x){
    if((rear+1)%N == front){
        cout << "overflow";
    }
    else if(front == -1 && rear == -1){
        front = rear = 0;
        queue[rear] = x;
    }
    else {
        rear = (rear+1)%N
        queue[rear] = x;
    }
}
void dequeu(){
 if(front == -1 && rear == -1){
    cout << "underflow";
 }
 else if(front == rear){
    cout << qua[front];
    front = rear = -1;
 }
```

```
else {
    cout << queue[front];
    front = (front + 1) % N;
}
}

void display() {
    int i = front;
    if (front == -1 && rear == -1) {
        cout << "empty";
    }
    else {
        while (i != rear) {
            cout << queue[i];
            i = (i+1) % N;
        }
        cout << queue[rear];
    }
}
```