04/11/25

Linear Queue

Pseudo Code :-

```
#define N 100
front = -1;
rear = -1;
Queue[N];

void enque(x):
    if (rear = N-1)
        print("Overflow")

    if (front == -1)
        front = rear = 0
        Queue[rear] = x
    else
        rear++;
        Queue[rear] = x;

void deque():
    if (front = rear = 0)
        print("Underflow")

    else
        print("Elemt xd is deleted", Queue[
        front++
```

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

int main () {
    int x;
    print(" 1. insert . 2. deletion 3. display 4. exit");
    ~~scanf("%d", &x)~~ : (x) supro
    ~~scan~~ while {
        ~~scan~~ print("Enter value");
        scanf("%d", &x);
        switch (x) :

            case 1:
                scanf( ___ )
                enque(n)

            case 2:
                deque()

            case 3:
                display()

            case 4:
                print("Exiting");
    } do (x != 4);

}}
```

Circular Queue

Pseudo Code :-

```
#define N 100
Circular Queue[N]
front = -1, rear = -1;

enque(x):
    if(front==0 and rear == N-1 (or)
                rear+1 = front)
        print("Over flow")
    else if(front == -1)
        front = 0, rear = 0
        Circular Queue[rear] = x
    else
        rear = (rear+1) % N
        Circular Queue[rear] = x

deque():
    if (front == -1):
        print("Underflow")
    ele if (front == rear):
        print("Deleted %d", Circular Queue[front])
        front = rear = -1
```

```
else : print("Deleted %d", CircularQueue[front])
       front = (front + 1) % N
```

Display():

```
if (front <= rear) {
        for (int i = front; i <= rear; i++)
            print(CQueue[i]);
}
else {
    for (int i = front; i <= N; i++)
        print(CircularQueue[i])

    for (int j = 0; j <= rear; j++)
        print(CircularQueue[j])
}
```

Nand
04/11

Linear Queue

Code ⇒

```
#include <stdio.h>
#include <ctype.h>
#define N 5
int front = -1, rear = -1;
int Queue[N];

void enque (int x) {
    if (rear == N-1) {
        printf ("Queue Overflow\n");
    }
    else if (front == -1 && rear == -1) {
        front = rear = 0;
        Queue [rear] = x;
    }
    else {
        Queue [++rear] = x;
    }
}

void deque() {
    if (front == -1 && rear == -1) {
        printf ("Queue Underflow\n");
    }
```

```
    else if (front == rear) {
        printf("Deleted element is %d\n", Queue[front]);
        front = rear = -1;
    }
    else {
        printf("Deleted element is %d\n", Queue[front++]);
}}

void display() {
    printf("Elements in the Queue are: ");
    for (int i=front; i<=rear; i++) {
        printf("%d\t", Queue[i]);
    }
    printf("\n");
}

int main() {
    int c;
    printf("1. Insert\n 2. delete\n 3. Disply\n 4. exit\n");
    do {
        printf("Choose an operation: ");
        scanf("%d", &c);
        switch(c) {
            case 1: {
                int x
                printf("Enter value to be inserted: ");
                scanf("%d", &x);
                enque(x);
            } break;
```

```
            case 2 : {
                        deque();
                        break;
                    }
            case 3 : {
                        display();
                        break;
                    }
            case 4 : {
                        printf("Exiting");
                        break;
                    }
            default : printf("Choose correct operation");
        }
    } while (c != 4);
}
```

Output :-

1. Insert
2. Deletion
3. Display
4. Exit

Enter a operation 1
Enter value to be inserted 1
Enter a operation 1
Enter value to be inserted 2
Enter operation : 3
Element in queue are 1 2
Enter operation 2
Deleted element is 1

Circular Queue

Code ⇒

```
#include <stdio.h>
#include <ctype.h>
# define N 5

int front = -1, rear = -1;
int Queue[N];

void enque(int x) {
    if ((front == 0 && rear == N-1) || front == rear+1) {
        printf("Queue Overflow\n");
    }
    else if (front == -1 && rear == -1) {
        front = rear = 0;
        Queue[rear] = x;
    }
    else {
        rear = (rear+1) % N;
        Queue[rear] = x;
    }
}

void deque() {
    if (front == -1 && rear == -1) {
        printf("Queue Underflow\n");
    }
    else if (front == rear) {
        printf("Deleted Element is %d\n", Queue[front]);
        front = rear = -1;
    }
}
```

```
    else {
        printf("Deleted Element is %d", Queue[front]);
        front = (front + 1) % N;
    }
}

void display() {
    printf("Elements are : ");
    if (front <= rear) {
        for (int i = front ; i <= rear ; i++) {
            printf("%d\t", Queue[i]);
        }
    }
    else {
        for (int i = front ; i <= N ; i++) {
            printf("%d\t", Queue[i]);
        }
        for (int i = 0 ; i <= rear ; i++) {
            printf("%d\t", Queue[i]);
        }
    }
}

int main() {
    int c;
    printf("1. Insert\n 2. Deletion\n 3. Display\n 4.
```

```
do {
    printf("Enter a operation:");
    scanf("%d", &c);
    switch (c) {
        case 1: {
            int x;
            printf("Enter the value to be inserted:");
            scanf("%d", &x);
            enque(x);
            break;
        }
        case 2: {
            deque();
            break;
        }
        case 3: {
            display();
            break;
        }
        case 4: {
            printf("Exiting");
            break;
        }
    }
} while (c != 4)
}
```

Output:

1. Insert
2. Deletion
3. Display
4. Exit

Enter a operation 1
Enter value to be inserted 1
Enter a operation 1
Enter value to be inserted 2
Enter a operation 1
Enter value to be inserted 3
Enter a operation 2
Deleted element is 1
Enter a operation 2
Deleted element is 2
Enter a operation 3
Elements in Queue are 3
Enter a operation 4
Exiting.

Rad
04/11