**AIM:** To implement a queue using array and linked list implementation and execute the following operation on stack

**ALGORITHM:**

1. **Algorithm to Enqueue an Element on to the Queue.**

**Step1:** if is ful()=true , go to step 2 , else go to step 3.

**Step2:** display a message a “Queue is over flow…!” and go to stop

**Step3:** set rear=rear+1

**Step4:** set queue[rear]=element

**Step5:** if front=-1, go to step 6 , else go to stop

**Step6:** set front=0

1. **Algorithm to Dequeue an Element from the Queue**

**Step 1** : Start.

**Step 2** : If IsEmpty() = True goto Step 3 else go to Step 4.

**Step 3** : Display message “Queue is Underflow…!” and goto Step 8.

**Step 4** : Display Queue[front].

**Step 5** : If front = rear goto Step 6 else goto Step 7.

**Step 6** : Set front = rear = -1 and goto Step 8.

**Step 7** : Set front = front + 1.

**Step 8** : Stop.

1. **Algorithm to Display Queue Elements Display()**

**Step 1 :** Start.

**Step 2 :** If IsEmpty() = True goto Step 3 else goto Step 4.

**Step 3 :** Display message “Queue is Underflow…!” and goto Step 8.

**Step 4 :** Set i = front.

**Step 5 :** Repeat Steps 6 to 7 while i <= rear.

**Step 6 :** Display Queue[i].

**Step 7 :** Set i = i + 1.

**Step 8 :** Stop.

**PROGRAM**

#include <stdio.h>

#define MAX 5

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

int IsFull();

int IsEmpty();

void Enqueue(int ele);

void Dequeue();

void Display();

int main()

{

int ch, e;

do {

printf("1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT");

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch(ch) {

case 1:

printf("Enter the element : ");

scanf("%d", &e);

Enqueue(e);

break;

case 2:

Dequeue();

break;

case 3:

Display();

break;

}

} while(ch <= 3);

return 0;

}

int IsFull()

{

if(rear == MAX - 1)

return 1;

else

return 0;

}

int IsEmpty()

{

if(front == -1)

return 1;

else

return 0;

}

void Enqueue(int ele)

{

if(IsFull())

printf("Queue is Overflow...!\n");

else{

rear = rear + 1;

Queue[rear] = ele;

if(front == -1)

front = 0;

}

}

void Dequeue()

{

if(IsEmpty())

printf("Queue is Underflow...!\n");

else{

printf("%d\n", Queue[front]);

if(front == rear)

front = rear = -1;

else

front = front + 1;

}

}

void Display()

{

int i;

if(IsEmpty())

printf("Queue is Underflow...!\n");

else {

for(i = front; i <= rear; i++)

printf("%d\t", Queue[i]);

printf("\n");

}

}

**SAMPLE OUTPUT**

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 10

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 20

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 30

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 2

10

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 3

20 30

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 4

**ALGORITHM:**

1. **Algorithm to Enqueue an Element on to the queue**

**Step 1 :** Start.

**Step 2 :** Set NewNode = addressof(Queue).

**Step 3 :** Set NewNode◊Element = e.

**Step 4 :** Set NewNode◊Next = NULL.

**Step 5 :** If Rear = NULL, then goto Step 6 else goto Step 7.

**Step 6 :** Set Front = Rear = NewNode and goto Step 9.

**Step 7 :** Set Rear◊Next = NewNode.

**Step 8 :** Set Rear = NewNode.

**Step 9 :** Stop.

1. **Algorithm to Dequeue an Element from the Queue**

**Step 1 :** Start.

**Step 2 :** If IsEmpty(Front) = True, then goto Step 3 else goto Step 4.

**Step 3 :** Display “Queue is Underflow…!” and goto Step 10.

**Step 4 :** Set TempNode = Front.

**Step 5 :** If Front = Rear goto Step 6 else goto Step 7.

**Step 6 :** Set Front = Rear = NULL and goto Step 8.

**Step 7 :** Set Front = Front◊Next.

**Step 8 :** Display TempNode◊Element.

**Step 9 :** Delete TempNode.

**Step 10:** Stop.

1. **Algorithm to Display Queue Elements**

**Step 1 :** Start.

**Step 2 :** If IsEmpty(Front) = TRUE goto Step 3 else goto Step 4.

**Step 3 :** Display “Queue is Underflow…!” and goto Step 8.

**Step 2:** Set Position = Front.

**Step 5 :** Repeat the Steps 6-7 until Position != NULL.

**Step 6 :** Display Position◊Element.

**Step 7 :** Set Position = Position◊Next.

**Step 8 :** Stop.

**PROGRAM**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int Element;

struct node \*Next;

}\*Front = NULL, \*Rear = NULL;

typedef struct node Queue;

int IsEmpty(Queue \*List);

void Enqueue(int e);

void Dequeue();

void Display();

int main()

{

int ch, e;

do{

printf("1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT");

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch(ch){

case 1:

printf("Enter the element : ");

scanf("%d", &e);

Enqueue(e);

break;

case 2:

Dequeue();

break;

case 3:

Display();

break;

}

} while(ch <= 3);

return 0;

}

int IsEmpty(Queue \*List)

{

if(List == NULL)

return 1;

else

return 0;

}

void Enqueue(int e){

Queue \*NewNode = malloc(sizeof(Queue));

NewNode->Element = e;

NewNode->Next = NULL;

if(Rear == NULL)

Front = Rear = NewNode;

else {

Rear->Next = NewNode;

Rear = NewNode;

}

}

void Dequeue()

{

if(IsEmpty(Front))

printf("Queue is Underflow...!\n");

else {

Queue \*TempNode;

TempNode = Front;

if(Front == Rear)

Front = Rear = NULL;

else

Front = Front->Next;

printf("%d\n", TempNode->Element);

free(TempNode);

}

}

void Display(){

if(IsEmpty(Front))

printf("Queue is Underflow...!\n");

else {

Queue \*Position;

Position = Front;

while(Position != NULL) {

printf("%d\t", Position->Element);

Position = Position->Next;

}

printf("\n");

}

}

**SAMPLE OUTPUT**

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 15

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 25

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 1

Enter the element : 35

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 2

15

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 3

25 35

1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT

Enter your choice : 4

**RESULT**

Hence the implementation of queue using array and linked list is executed