```
2.Pop

3.Exit

2

Stack is emptySTACK :

1.Push

2.Pop

3.Exit

2

Stack is emptySTACK :

1.Push

2.Pop

3.Exit

3
```

a/1/22

Queue using Linked List

Aim

Implement a queue using linked list witho operatrons. 1. Insert an elements to the queue 2. Delete element from the queue 3. Display the queue after each operation.

Algorithm

```
step 1: Start
step 2: De a structure node that contains.
        1) Int data
        2) pointer to struct node, link.
step 3: Declare variables of node, *front, * rear.
step 4: declare a function calle enque.
        1) Allocte memory to temp.
        2) ~~a~~dd the data to temp->dat.
        3) make temp->link to NULL
        4). if front == NULL
                front = rear = temp.
        5). else
              rear->link = temp.
                rear = temp.
```

```
step 5: Inside the function deque.
    1) if front == NULL das
       display that the queue is empty.
    2). else,
        temp = front.
        front = front ->link.
        and free the memory of temp.

step 6: Inside the function display.
    1) if front == NULL.
       print que is empty
    2) else temp = front
       while (tem != NULL)
           display tem->ddt
           temp = tem->ddn

step 7: Stop.
```

Result
The program is Executed an output is verified.

Code

```c
#include<stdio.h>
#include<stdlib.h>


struct node
{
    int data;
    struct node* link;
};

struct node* head=NULL;


struct node* front=NULL;
struct node* rear=NULL;


// void display()
// {
//   struct node* temp=head;
//   while(temp!=NULL)
//   {
//       printf("%d ",temp->data);
//       temp=temp->link;
//   }
//   printf("\n");
// }


void enqueue(int data)
{
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->link=NULL;
    if(front==NULL || rear==NULL)
    {
        front=rear=newnode;
    }else
    {
        rear->link=newnode;
        rear=newnode;
    }
}

int dequeue()
{
    if(front==NULL || rear==NULL)
    {
        printf("Queue is empty");
        return -1;
    }else
    {
        int val=front->data;
        front=front->link;
        return val;
    }
}

void display()
{
    printf("QUEUE:\n");
    struct node* temp=front;
    while(temp!=NULL)
    {
        printf("%d ",temp->data);
        temp=temp->link;
    }
    printf("\n");
}

// void enqueue(int data)
// {
//  struct node* newnode=(struct node*)malloc(sizeof(struct node));
//  newnode->data=data;
//  newnode->link=head;
//  head=newnode;
// }





// int dequeue()
// {
//  struct node* temp=head;
//  int val=-1;
//  if(temp==NULL)
//  {
//      printf("The Queue is empty\n");
//  }else
//  {
//      if(head->link!=NULL)
//      {
//          while(temp->link->link!=NULL)
//          {
//              temp=temp->link;
//          }
//          val=temp->link->data;
//          temp->link=NULL;

//      }else
//      {
//          val=head->data;
//          head=NULL;
//      }
//  }
//  return val;
// }



int main()
{



    while(1)
    {




        int choice;
        printf("1.Enqueue\n2.Dequeue\n3.Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {

            case 1:
            {
                int temp;
                printf("Enter the number to enqueue");
                scanf("%d",&temp);
                enqueue(temp);
                display();
                break;
            }
            case 2:
            {
                int temp=dequeue();
                if(temp!=-1)
                {
                    printf("%d Dequeued\n",temp);
                }
                display();
                break;
            }
            case 3:
            {
                return 0;
            }
        }
    }
    return 0;
}
```

OUTPUT

```
1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue12

QUEUE:

12

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue13

QUEUE:

12 13

1.Enqueue

2.Dequeue

3.Exit

1

Enter the number to enqueue14

QUEUE:

12 13 14

1.Enqueue

2.Dequeue
```

```
3.Exit

1

Enter the number to enqueue15

QUEUE:

12 13 14 15

1.Enqueue

2.Dequeue

3.Exit

2

12 Dequeued

QUEUE:

13 14 15

1.Enqueue

2.Dequeue

3.Exit

2

13 Dequeued

QUEUE:

14 15

1.Enqueue

2.Dequeue

3.Exit

2

14 Dequeued

QUEUE:

15

1.Enqueue

2.Dequeue

3.Exit

2

15 Dequeued

QUEUE:
```

```
1.Enqueue

2.Dequeue

3.Exit

2

Queue is emptyQUEUE:


1.Enqueue

2.Dequeue

3.Exit

3
```