# EXPERIMENT – 3.1

**Name:** Rohan Jaiswal
**UID:** 21BCS2856
**Branch:** CSE
**Section/Group:** 608 (B)
**Semester:** 3rd
**Date of Performance:** 13th Oct
**Subject Name:** DS
**Subject Code:** 21CSH-211

## Aim of Practical:

Write a program to demonstrate the implementation of various operations on a linear queue and circular represented using a linear linked list.

## Algorithm:

1. If rear=max
   print overflow
   Exit;
2. If front=rear=-1
   then set front=rear=0;
   else
   rear++;
3. Queue[rear] = item
4. Exit

## Program Code:

```
#include<iostream>
using namespace std;

struct Node{
    int val;
    struct Node* next;
    Node(int data){
        val = data;
        next=NULL;
    }
};

struct Queue{
    Node* front;
    Node* rear;
    int capacity;
    int size=0;

    Queue(int s){
        capacity=s;
        front=NULL;
        rear=NULL;
    }

    void display(){
        if(!front){
            cout<<"Queue is Empty!\n";
            return;
        }

        Node* itr=front;
        cout<<"Queue Elements: ";
        while(itr){
            cout<<itr->val<<" ";
            itr = itr->next;
        }
        cout<<"\n";
    }
```

```
    void getFront(){
        if(!front){
            cout<<"Queue is Empty!\n";
            return;
        }

        cout<<"Front Element: "<<front->val<<"\n";
    }

    void push(int data){
        Node* tmp = new Node(data);
        if(size>=capacity){
            cout<<"Queue is Full!\n";
            return;
        }

        if(!front){
            front=tmp;
            rear=tmp;
        }
        else{
            rear->next=tmp;
            rear=tmp;
        }
        size++;
    }

    void pop(){
        if(!front){
            cout<<"Queue is Empty!\n";
            return;
        }

        Node* tmp = front;
        front = front->next;
        delete tmp;
        size--;
    }
};
```

```
int main(){
    cout<<"Enter Size of the Queue: ";
    int size;cin>>size;

    Queue q(size);

    bool flag = true;
    while(flag){
        std::cout<<"\nQueue basic operation Menu :-\n";
        std::cout<<"1. Display\n2. Get Front\n3. push.\n4. Delete.\n5. Exit
Program\n\n";
        std::cout<<"Your choice: ";
        std::string choice; std::cin>>choice;

        if(choice.size()>1) // for tackling when input is alphabet and strings.
            choice[0]='6';

        switch (choice[0])
        {
        case '1':
            q.display();
            break;
        case '2':
            q.getFront();
            break;
        case '3':
            std::cout<<"Enter the value of Element you want to push: ";
            int val;std::cin>>val;
            q.push(val);
            break;
        case '4':
            q.pop();
            break;
        case '5':
            flag = false;
            std::cout<<"Exiting.......";
            break;
        default:
            std::cout<<"Invalid Choice... try again!";
            break;
        }
```

```
            std::cout<<"\n";
            system("pause");
            std::cout << "\033[2J\033[1;1H"; //for clearing screen in terminal.
        }
        std::cout<<"Program Stopped!!";
    }
```

## Output:

```
Enter Size of the Queue: 2

Queue basic operation Menu :-
1. Display
2. Get Front
3. push.
4. Delete.
5. Exit Program

Your choice: 3
Enter the value of Element you want to push: 5

Press any key to continue . . .
```

```
Queue basic operation Menu :-
1. Display
2. Get Front
3. push.
4. Delete.
5. Exit Program

Your choice: 3
Enter the value of Element you want to push: 7

Press any key to continue . . .
```

```
Queue basic operation Menu :-
1. Display
2. Get Front
3. push.
4. Delete.
5. Exit Program

Your choice: 1
Queue Elements: 5 7

Press any key to continue . . .
```

```
Queue basic operation Menu :-
1. Display
2. Get Front
3. push.
4. Delete.
5. Exit Program

Your choice: 2
First Element: 5

Press any key to continue . . .
```