QUEUE USING SINGLY LINKED LIST :

```
#include<iostream>

using namespace std;

class Node {
  public:
   int key;
        int data; // value
        Node * next;

  Node() {
    key = 0;
    data = 0;
    next = NULL;
  }
  Node(int k, int d) {
    key = k;
    data = d;
    next = NULL;
  }
};

class Queue
{
  public:
        Node *front;
        Node *rear;

    Queue()
         {
      front = NULL;
      rear = NULL;
    }

    bool isEmpty()
    {
        if(front==NULL && rear==NULL)
        {
                return true;
                }
                else
                {
                        return false;
                }
         }

         bool checkIfNodeExist(Node *n)
         {
                Node *temp = front;
                bool exist=false;
                while(temp!=NULL)
                {
                        if(temp->key==n->key)
        {
          exist=true;
          break;
        }
        temp=temp->next;
                }
                return exist;
         }

        void enqueue(Node *n)
  {
   if (isEmpty())
     {
       front = n;
       rear = n;
       cout<<"Node  ENQUEUED successfully"<<endl;
     }
   else if(checkIfNodeExist(n))
   {
     cout<<"Node already exist with this Key value."
     <<"Enter different Key value"<<endl;
   }
   else
   {
     rear->next=n;
     rear=n;
     //top = n;
     cout<<"Node  ENQUEUED successfully"<<endl;
   }

  }

        Node* dequeue()
   {
       Node *temp=NULL;
     if (isEmpty())
     {
         cout << "Queue is Empty" << endl;
         return NULL;
     }
     else
     {
       if(front==rear)
       {
         temp=front;
         front = NULL;
         rear = NULL;
         return temp;
       }
       else
       {
         temp=front;
```

```
          front = front->next;
          return temp;
        }

      }
    }

         int count()
    {
      int count=0;
      Node *temp=front;
      while(temp!=NULL)
      {
        count++;
        temp=temp->next;
        }
     return count;
    }

         void display()
    {
      if(isEmpty())
      {
        cout << "Queue is Empty" << endl;
      }
    else
    {
      cout << "All values in the Queue are :" << endl;
        Node *temp=front;
        while(temp!=NULL)
        {
          cout<<"("<<temp->key<<","<<temp->data<<")"<<" -> ";
          temp=temp->next;
         }
      cout<<endl;
    }

    }


};

int main() {
  Queue q;
  int option,key, data;

  do {
    cout << "What operation do you want to perform?"
         <<"Select Option number. Enter 0 to exit." << endl;
    cout << "1. Enqueue()" << endl;
    cout << "2. Dequeue()" << endl;
    cout << "3. isEmpty()" << endl;
    cout << "4. count()" << endl;
    cout << "5. display()" << endl;
cout << "6. Clear Screen" << endl << endl;
      cin >> option;

       //Node n1 = new Node();
      Node * new_node = new Node();


switch (option) {
case 0:
  break;
case 1:
     cout << "ENQUEUE Function Called -" <<endl;
  cout << "Enter KEY and VALUE of NODE to ENQUEUE "
             <<"in the Queue"
             <<endl;
  cin >> key;
  cin >> data;
  new_node->key = key;
  new_node->data = data;
  q.enqueue(new_node);
  break;
case 2:
  cout << "DEQUEUE Function Called - " <<endl;
  new_node = q.dequeue();
  cout<<"Dequeued Value is: ("<<new_node->key<<","
             <<new_node->data<<")";
  delete new_node;
             cout<<endl;

  break;
case 3:
     cout << "isEmpty Function Called - " << endl;
  if (q.isEmpty())
    cout << "Queue is Empty" << endl;
  else
    cout << "Queue is NOT Empty" << endl;
  break;
case 4:
     cout << "Count Function Called - " << endl;
  cout << "No of nodes in the Queue: " <<q.count()
             <<endl;
  break;
case 5:
  cout << "Display Function Called - " << endl;
  q.display();
  cout << endl;
  break;

case 6:
  system("cls");
  break;
default:
  cout << "Enter Proper Option number " << endl;
}
```

```
  } while (option != 0);

  return 0;
}
```