

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2024-25**

**Data Structures – Lab Batch 2023-27**

|  |  |
| --- | --- |
| **Lab Assignment No:- 7** | |
|  | |
| **Name of Student** | Pankhuri Varshney |
| **PRN No.** | 23070122160 |
| **Batch** | 2023-27 |
| **Class** | CS-B2 |
| **Academic Year & Semester** | 2024-25  Semester 3 |
| **Date of Performance** | 2st October, 2024 |
|  | |
| **Title of Assignment:** | * Implement Stack using Array * Implement Stack using Linked List * Implement Queue using Array * Implement Queue using Linked List |
| **Source Code/Algorithm/Flow Chart:** | **Write programs to:**   1. **Implement Stack using Array** 2. **Implement Stack using Linked List** 3. **Implement Queue using Array** 4. **Implement Queue using Linked List**   **SOURCE CODE:**   1. **Implement Stack using Array** 2. **Implement Stack using Linked List**   **SOURCE CODE:**   1. **Implement Queue using Array**   **SOURCE CODE:**  #include <iostream>  using namespace std;  class Queue{  int MAX\_SIZE;  int front;  int rear;  int \*queue;  public:  //constructor:  Queue(int size) : MAX\_SIZE(size), front(-1), rear(-1) {  queue = new int[MAX\_SIZE];  }  //deconstructor:  ~Queue(){  free(queue);  }  //func to add nodes from rear end  void enqueue(int val){  if(rear==MAX\_SIZE-1){  cout<<"QUEUE OVERFLOW!"<<endl;  return;  }  if(rear==-1){  front=0;  }  queue[++rear]=val;  cout<<"ADDED VAL: "<<val<<endl;  }  //func to remove nodes from front end  void dequeue(){  if(front==-1){  cout<<"QUEUE UNDERFLOW"<<endl;  return;  }  cout<<"DELETED VAL: "<<queue[front]<<endl;  if(front==rear){  queue[front]=0;  front=rear=-1;  return;  }  queue[front++]=0;  }  //func to display entire queue  void display(){  if(front>rear || front==-1){  cout<<"EMPTY QUEUE"<<endl;  return;  }  for(int i=front; i<=rear; i++){  cout<<queue[i]<<"\t";  }  cout<<endl;  }  };  int main(){  cout<<"QUEUE USING ARRAYS: \nEnter Max Size: ";  int size;  cin>>size;  Queue q(size);  q.enqueue(10);  q.display();  q.enqueue(20);  q.display();  q.enqueue(30);  q.display();  q.enqueue(40);  q.display();  q.dequeue();  q.display();  q.dequeue();  q.display();  q.dequeue();  q.display();  q.dequeue();  q.display();  return 0;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. **Implement Queue using Linked List**   **SOURCE CODE:**  #include <iostream>  using namespace std;  class node{  public:  int data;  node \*link;  node(int data){  this->data=data;  this->link=nullptr;  }  };  class LinkedList {  public:  node \*head;  public:  LinkedList() {  this->head = nullptr;  }  ~LinkedList() {  node \*current = head;  while (current != nullptr) {  node \*next = current->link;  delete current;  current = next;  }  }  void addAtEnd(int data) {  node \*newNode = new node(data);  if (head == nullptr) {  head = newNode;  } else {  node \*temp = head;  while (temp->link != nullptr) {  temp = temp->link;  }  temp->link = newNode;  }  }  void displayList() {  node \*temp = head;  while (temp != nullptr) {  std::cout << temp->data <<"\t";  temp = temp->link;  }  cout << endl;  }  };  class queue{  private:  LinkedList &q;  node \*front;  node \*rear;    public:  queue(LinkedList &q) : q(q) {  front = q.head;  if (front == nullptr) {  rear = nullptr;  } else {  node \*temp = front;  while (temp->link != nullptr) {  temp = temp->link;  }  rear = temp;  }  }    void enqueue(int data){  node \*newNode = new node(data);  if (rear != nullptr) {  rear->link = newNode;  rear = newNode;  } else {  front = rear = newNode;  }  }  void dequeue(){  if(front == nullptr){  cout<<"QUEUE UNDERFLOW\n";  return;  }  else{  node \*temp=front;  front=front->link;  delete temp;  if(front == nullptr) {  rear = nullptr;  }  }  }  };  int main(){  LinkedList li;  li.addAtEnd(10);  li.addAtEnd(20);  li.addAtEnd(30);  li.addAtEnd(40);  li.addAtEnd(50);  queue qu(li);  li.displayList();  qu.enqueue(60);  li.displayList();  qu.dequeue();  li.displayList();  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Output Screenshots** |  |
| **Practice questions** |  |
| **Conclusion** | Thus, we have studied the concept of Linked List and how it is different from arrays. |