Name : [AZHAR Ali]

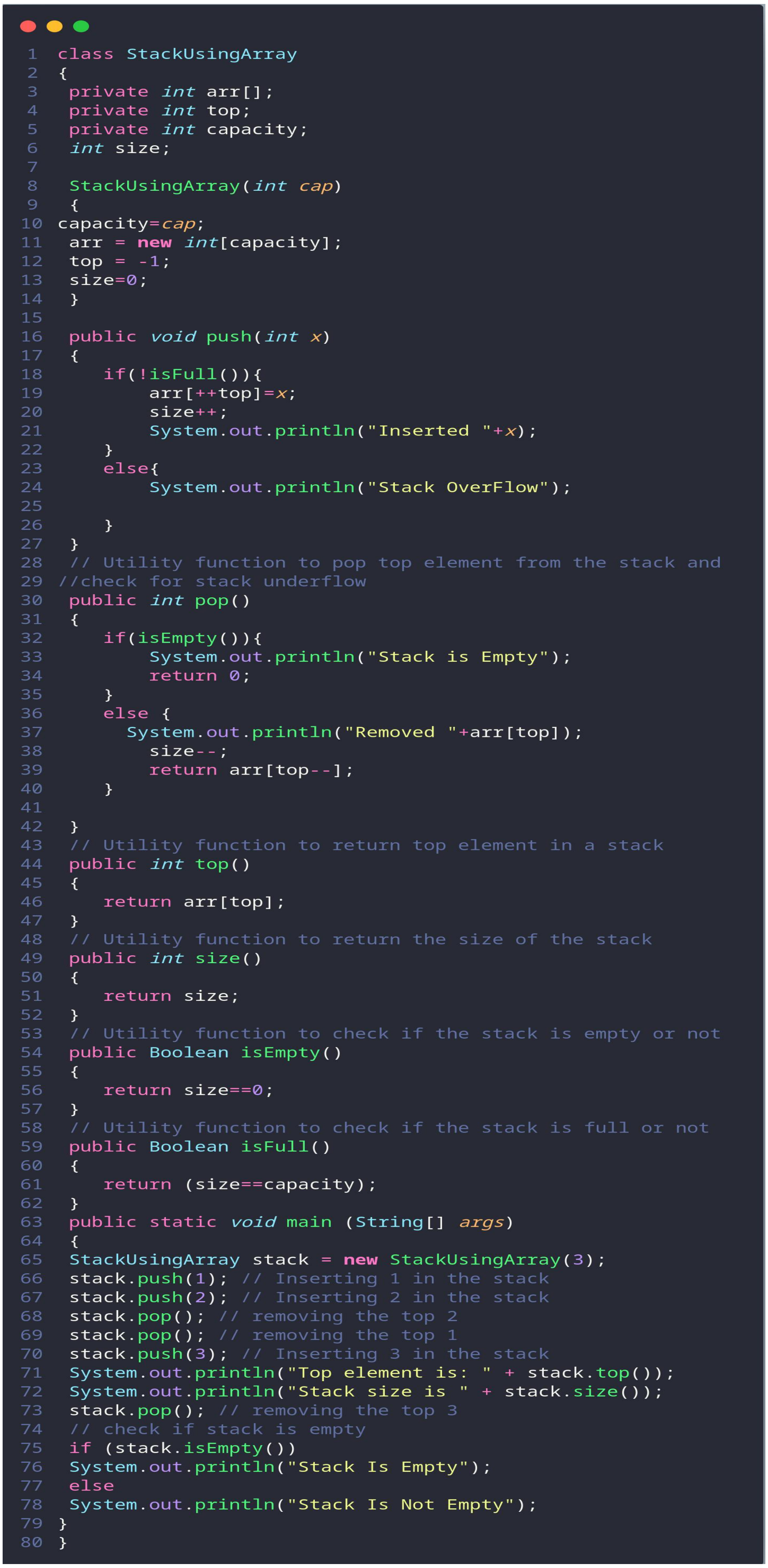
CMS : [023-23-0314]

SEC : [A]

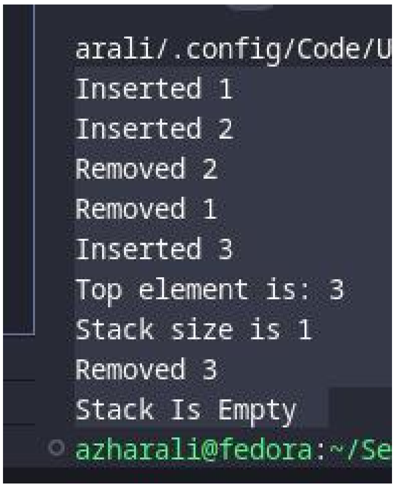
LAB 4

DSA LAB

1. **Stack using array**: Understand provided code and implement all required
methods in Stack. Stack Code is given below:

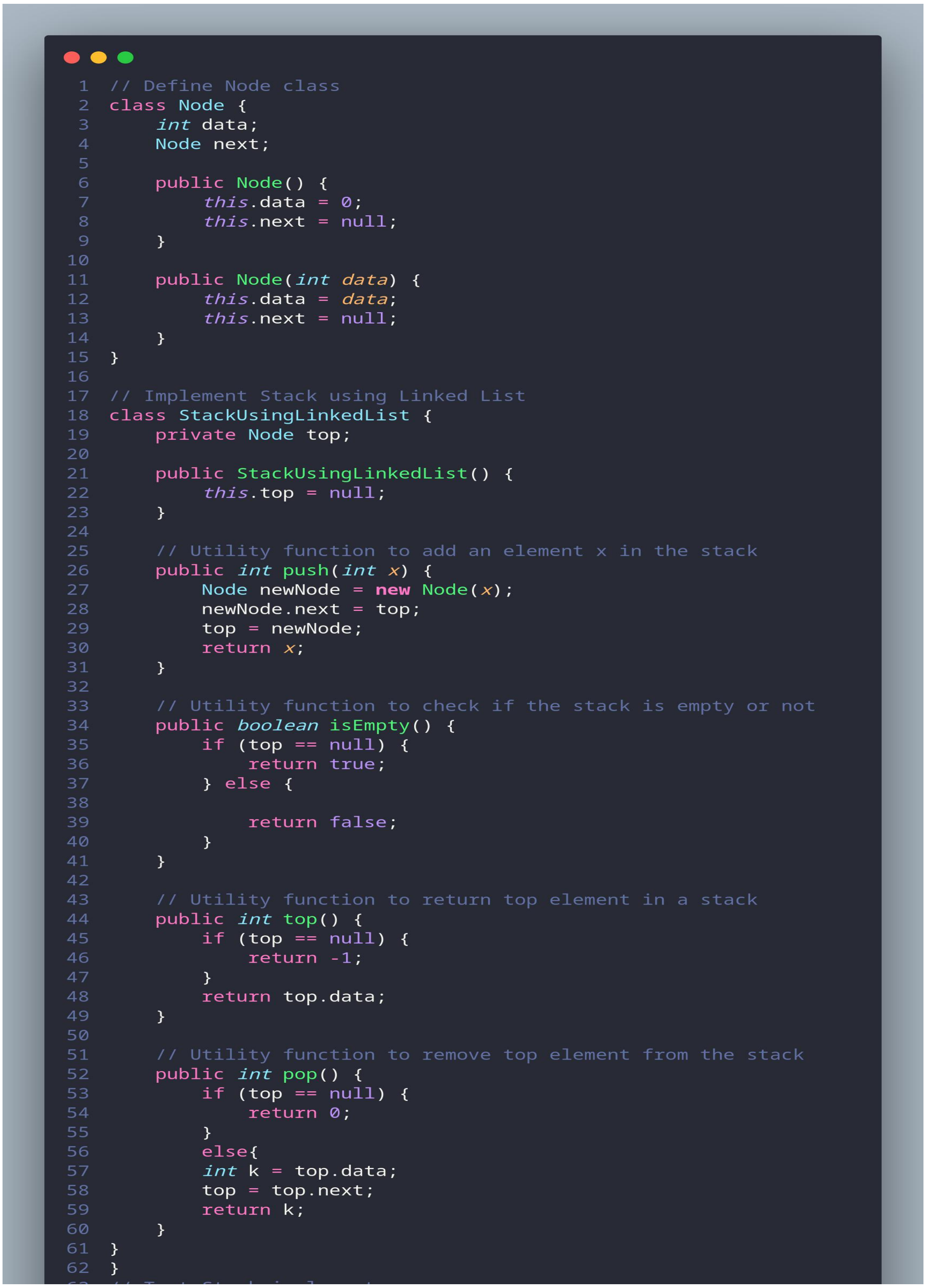


```java
class StackUsingArray
{
 private int arr[];
 private int top;
 private int capacity;
 int size;

 StackUsingArray(int cap)
 {
capacity=cap;
 arr = new int[capacity];
 top = -1;
 size=0;
 }

 public void push(int x)
 {
    if(!isFull()){
        arr[++top]=x;
        size++;
        System.out.println("Inserted "+x);
    }
    else{
        System.out.println("Stack OverFlow");

    }
 }
 // Utility function to pop top element from the stack and
//check for stack underflow
 public int pop()
 {
    if(isEmpty()){
        System.out.println("Stack is Empty");
        return 0;
    }
    else {
      System.out.println("Removed "+arr[top]);
        size--;
        return arr[top--];
    }

 }
 // Utility function to return top element in a stack
 public int top()
 {
    return arr[top];
 }
 // Utility function to return the size of the stack
 public int size()
 {
    return size;
 }
 // Utility function to check if the stack is empty or not
 public Boolean isEmpty()
 {
    return size==0;
 }
 // Utility function to check if the stack is full or not
 public Boolean isFull()
 {
    return (size==capacity);
 }
 public static void main (String[] args)
 {
 StackUsingArray stack = new StackUsingArray(3);
 stack.push(1); // Inserting 1 in the stack
 stack.push(2); // Inserting 2 in the stack
 stack.pop(); // removing the top 2
 stack.pop(); // removing the top 1
 stack.push(3); // Inserting 3 in the stack
 System.out.println("Top element is: " + stack.top());
 System.out.println("Stack size is " + stack.size());
 stack.pop(); // removing the top 3
 // check if stack is empty
 if (stack.isEmpty())
 System.out.println("Stack Is Empty");
 else
 System.out.println("Stack Is Not Empty");
}
}
```



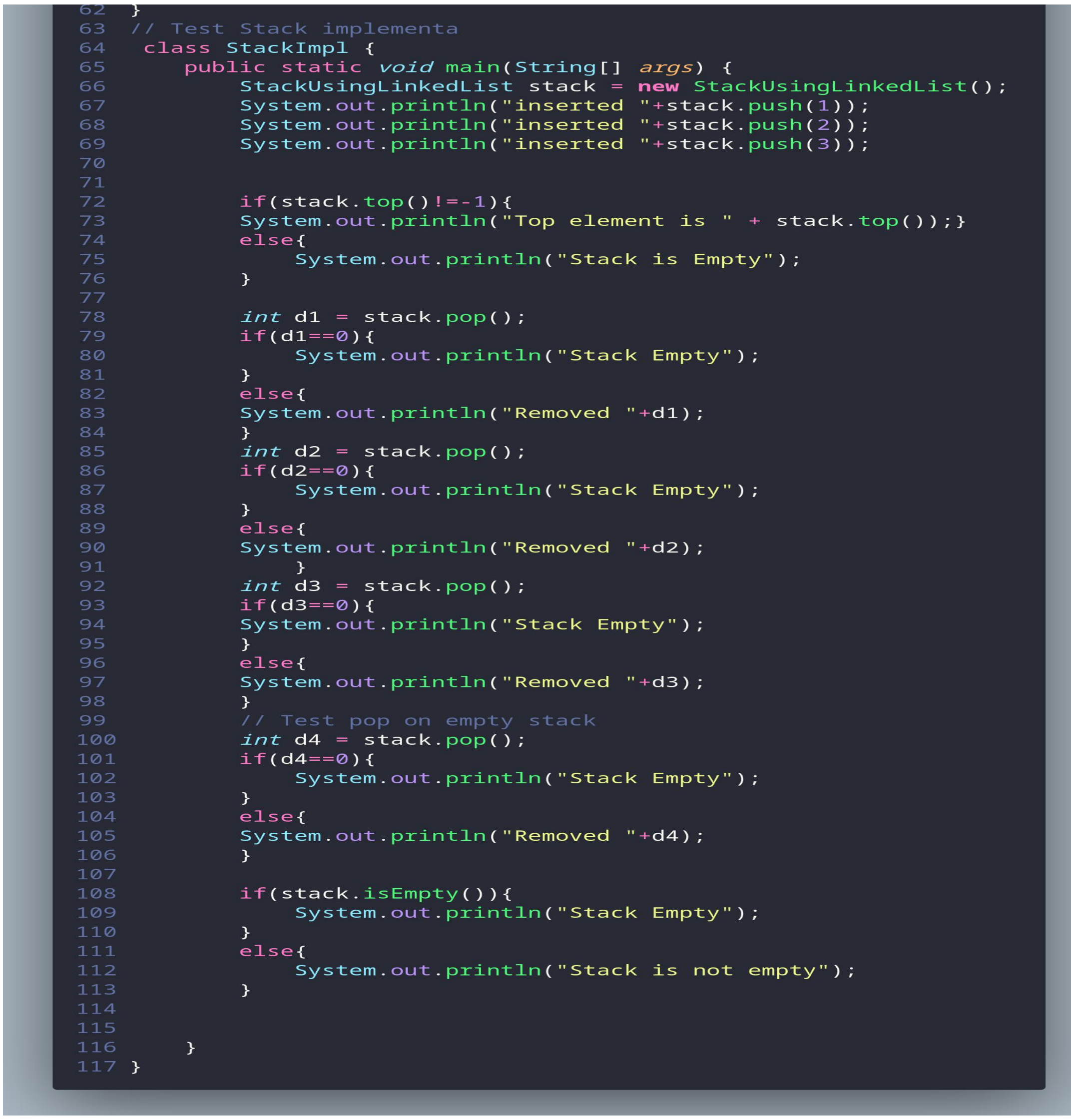
```
arali/.config/Code/U
Inserted 1
Inserted 2
Removed 2
Removed 1
Inserted 3
Top element is: 3
Stack size is 1
Removed 3
Stack Is Empty
azharali@fedora:~/Se
```

2. **Stack using Linked list**: Understand provided code and implement all required
methods in Stack. Stack Code is given below:

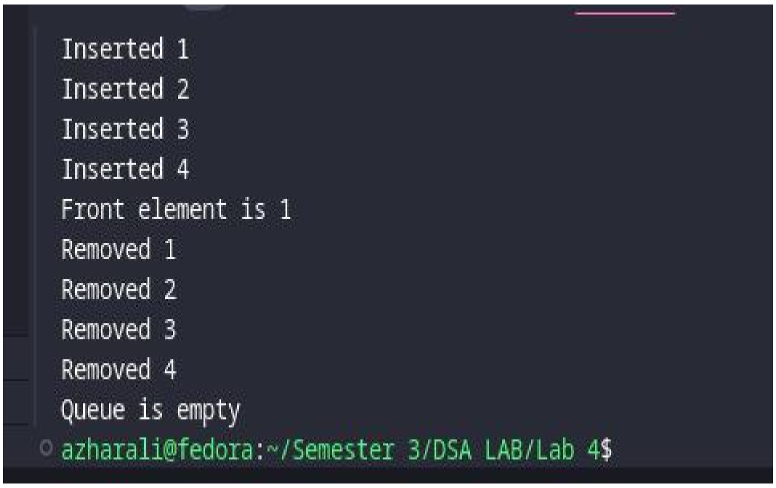


```
// Define Node class
class Node {
    int data;
    Node next;

    public Node() {
        this.data = 0;
        this.next = null;
    }

    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}

// Implement Stack using Linked List
class StackUsingLinkedList {
    private Node top;

    public StackUsingLinkedList() {
        this.top = null;
    }

    // Utility function to add an element x in the stack
    public int push(int x) {
        Node newNode = new Node(x);
        newNode.next = top;
        top = newNode;
        return x;
    }

    // Utility function to check if the stack is empty or not
    public boolean isEmpty() {
        if (top == null) {
            return true;
        } else {

            return false;
        }
    }

    // Utility function to return top element in a stack
    public int top() {
        if (top == null) {
            return -1;
        }
        return top.data;
    }

    // Utility function to remove top element from the stack
    public int pop() {
        if (top == null) {
            return 0;
        }
        else{
        int k = top.data;
        top = top.next;
        return k;
    }
}
}
```

df



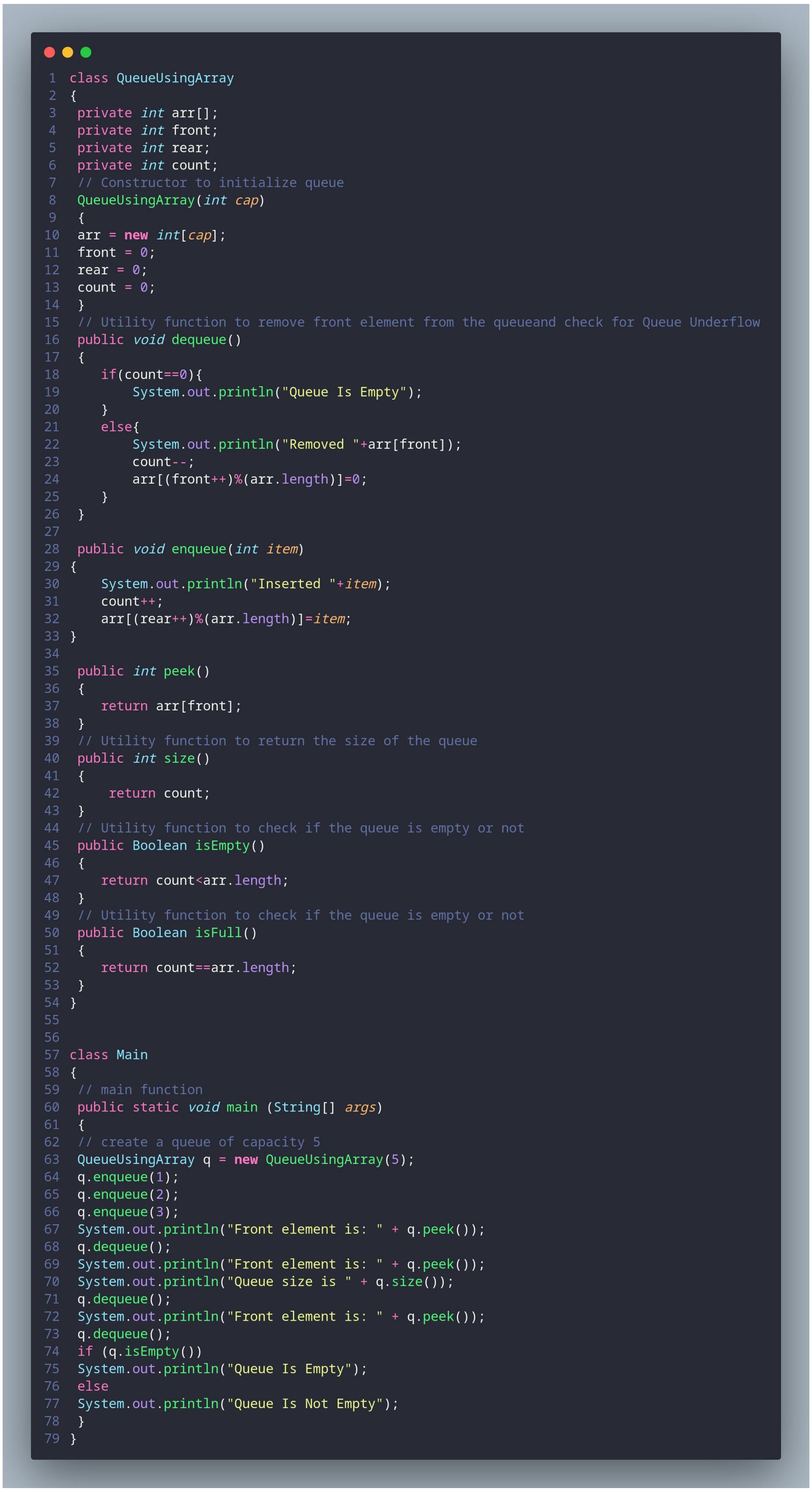
```
}
// Test Stack implementa
 class StackImpl {
    public static void main(String[] args) {
        StackUsingLinkedList stack = new StackUsingLinkedList();
        System.out.println("inserted "+stack.push(1));
        System.out.println("inserted "+stack.push(2));
        System.out.println("inserted "+stack.push(3));


        if(stack.top()!=-1){
        System.out.println("Top element is " + stack.top());}
        else{
            System.out.println("Stack is Empty");
        }

        int d1 = stack.pop();
        if(d1==0){
            System.out.println("Stack Empty");
        }
        else{
        System.out.println("Removed "+d1);
        }
        int d2 = stack.pop();
        if(d2==0){
            System.out.println("Stack Empty");
        }
        else{
        System.out.println("Removed "+d2);
            }
        int d3 = stack.pop();
        if(d3==0){
        System.out.println("Stack Empty");
        }
        else{
        System.out.println("Removed "+d3);
        }
        // Test pop on empty stack
        int d4 = stack.pop();
        if(d4==0){
            System.out.println("Stack Empty");
        }
        else{
        System.out.println("Removed "+d4);
        }

        if(stack.isEmpty()){
            System.out.println("Stack Empty");
        }
        else{
            System.out.println("Stack is not empty");
        }


    }
}
```



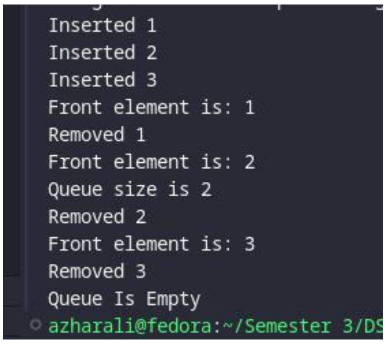
```
Inserted 1
Inserted 2
Inserted 3
Inserted 4
Front element is 1
Removed 1
Removed 2
Removed 3
Removed 4
Queue is empty
azharali@fedora:~/Semester 3/DSA LAB/Lab 4$
```

df

3. **Queue using array**: Understand provided code and implement all required
methods in Queue. Queue Code is given below:



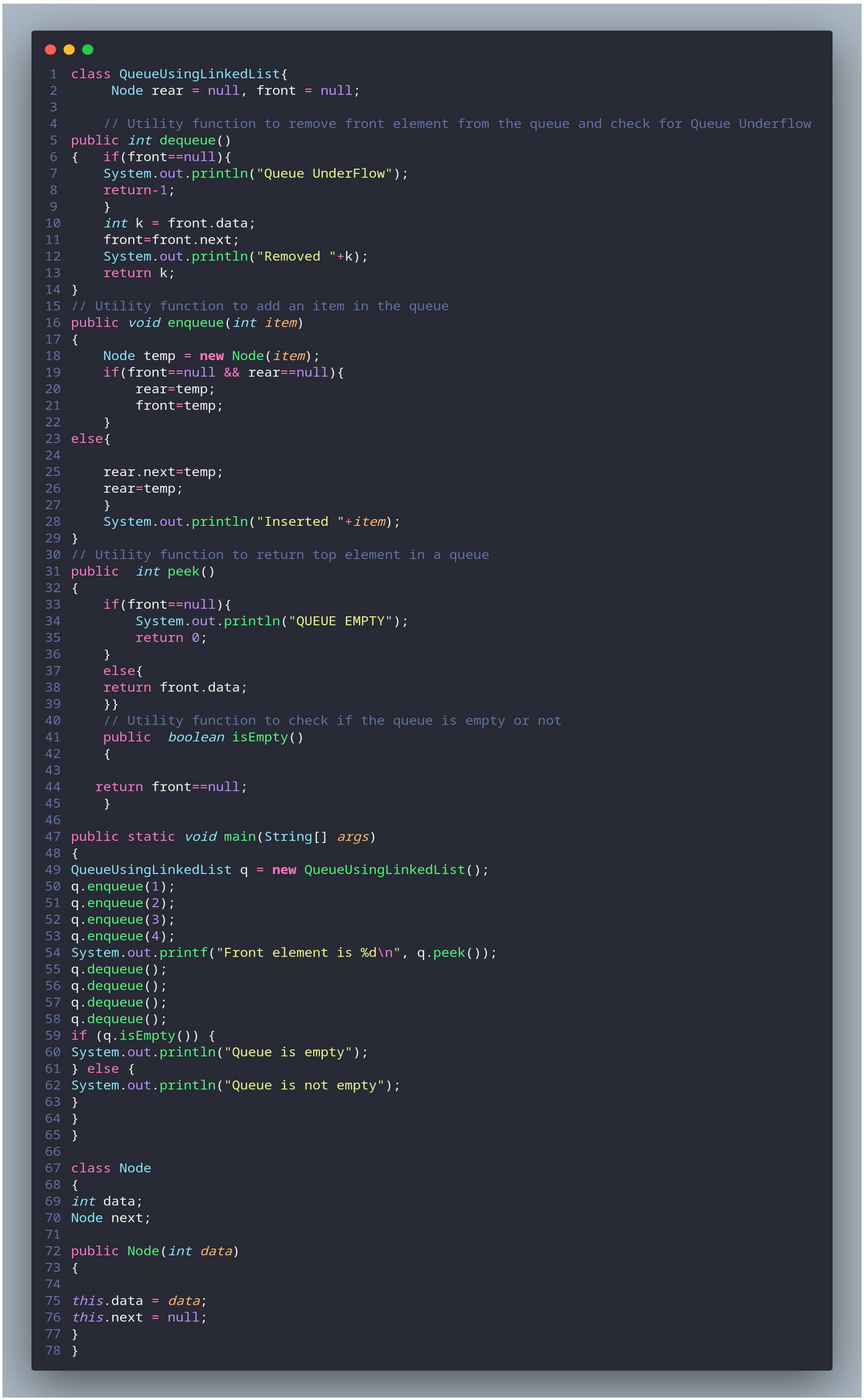
```java
class QueueUsingArray
{
 private int arr[];
 private int front;
 private int rear;
 private int count;
 // Constructor to initialize queue
 QueueUsingArray(int cap)
 {
 arr = new int[cap];
 front = 0;
 rear = 0;
 count = 0;
 }
 // Utility function to remove front element from the queueand check for Queue Underflow
 public void dequeue()
 {
    if(count==0){
        System.out.println("Queue Is Empty");
    }
    else{
        System.out.println("Removed "+arr[front]);
        count--;
        arr[(front++)%(arr.length)]=0;
    }
 }

 public void enqueue(int item)
{
    System.out.println("Inserted "+item);
    count++;
    arr[(rear++)%(arr.length)]=item;
}

 public int peek()
 {
    return arr[front];
 }
 // Utility function to return the size of the queue
 public int size()
 {
     return count;
 }
 // Utility function to check if the queue is empty or not
 public Boolean isEmpty()
 {
    return count<arr.length;
 }
 // Utility function to check if the queue is empty or not
 public Boolean isFull()
 {
    return count==arr.length;
 }
}


class Main
{
 // main function
 public static void main (String[] args)
 {
 // create a queue of capacity 5
 QueueUsingArray q = new QueueUsingArray(5);
 q.enqueue(1);
 q.enqueue(2);
 q.enqueue(3);
 System.out.println("Front element is: " + q.peek());
 q.dequeue();
 System.out.println("Front element is: " + q.peek());
 System.out.println("Queue size is " + q.size());
 q.dequeue();
 System.out.println("Front element is: " + q.peek());
 q.dequeue();
 if (q.isEmpty())
 System.out.println("Queue Is Empty");
 else
 System.out.println("Queue Is Not Empty");
 }
}
```



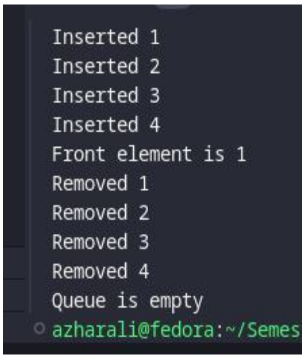
```
Inserted 1
Inserted 2
Inserted 3
Front element is: 1
Removed 1
Front element is: 2
Queue size is 2
Removed 2
Front element is: 3
Removed 3
Queue Is Empty
○ azharali@fedora:~/Semester 3/DS
```

df

4. **Queue using Linked list**: Understand provided code and implement all required methods in Queue. Queue Code is given below:



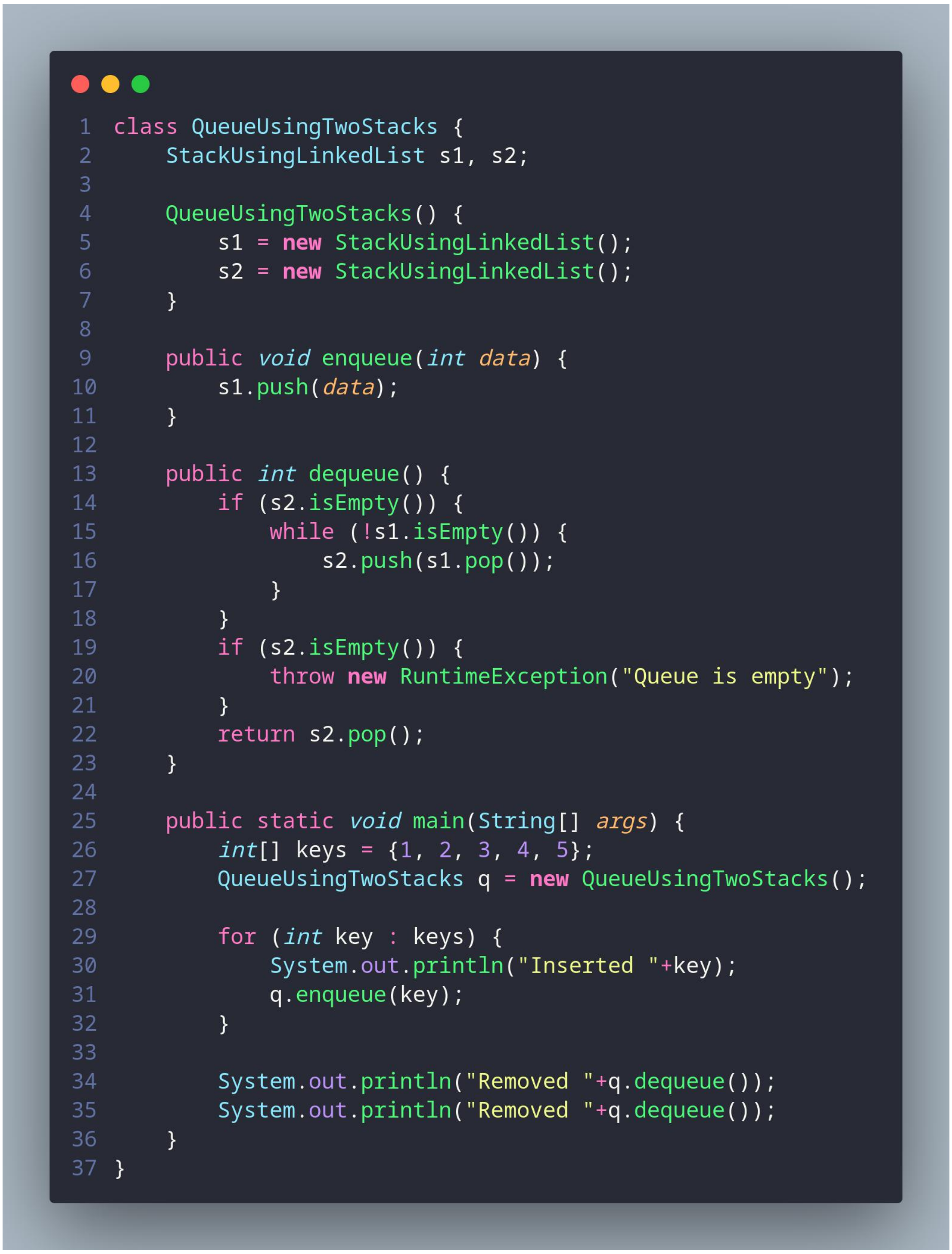
```java
class QueueUsingLinkedList{
     Node rear = null, front = null;

    // Utility function to remove front element from the queue and check for Queue Underflow
public int dequeue()
{   if(front==null){
    System.out.println("Queue UnderFlow");
    return-1;
    }
    int k = front.data;
    front=front.next;
    System.out.println("Removed "+k);
    return k;
}
// Utility function to add an item in the queue
public void enqueue(int item)
{
    Node temp = new Node(item);
    if(front==null && rear==null){
        rear=temp;
        front=temp;
    }
else{

    rear.next=temp;
    rear=temp;
    }
    System.out.println("Inserted "+item);
}
// Utility function to return top element in a queue
public  int peek()
{
    if(front==null){
        System.out.println("QUEUE EMPTY");
        return 0;
    }
    else{
    return front.data;
    }}
    // Utility function to check if the queue is empty or not
    public  boolean isEmpty()
    {

   return front==null;
    }

public static void main(String[] args)
{
QueueUsingLinkedList q = new QueueUsingLinkedList();
q.enqueue(1);
q.enqueue(2);
q.enqueue(3);
q.enqueue(4);
System.out.printf("Front element is %d\n", q.peek());
q.dequeue();
q.dequeue();
q.dequeue();
q.dequeue();
if (q.isEmpty()) {
System.out.println("Queue is empty");
} else {
System.out.println("Queue is not empty");
}
}
}

class Node
{
int data;
Node next;

public Node(int data)
{

this.data = data;
this.next = null;
}
}
```



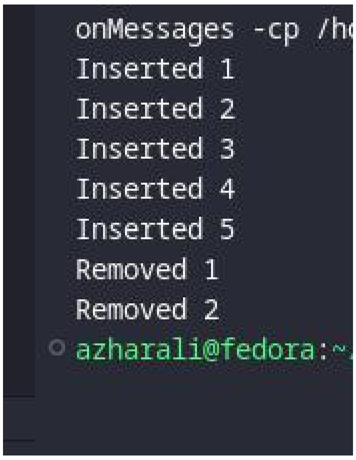
```
Inserted 1
Inserted 2
Inserted 3
Inserted 4
Front element is 1
Removed 1
Removed 2
Removed 3
Removed 4
Queue is empty
azharali@fedora:~/Semes
```

df

5. **Queue using two Stacks**: Understand provided code and implement all required methods in Queue Class. Sample Code is given below:



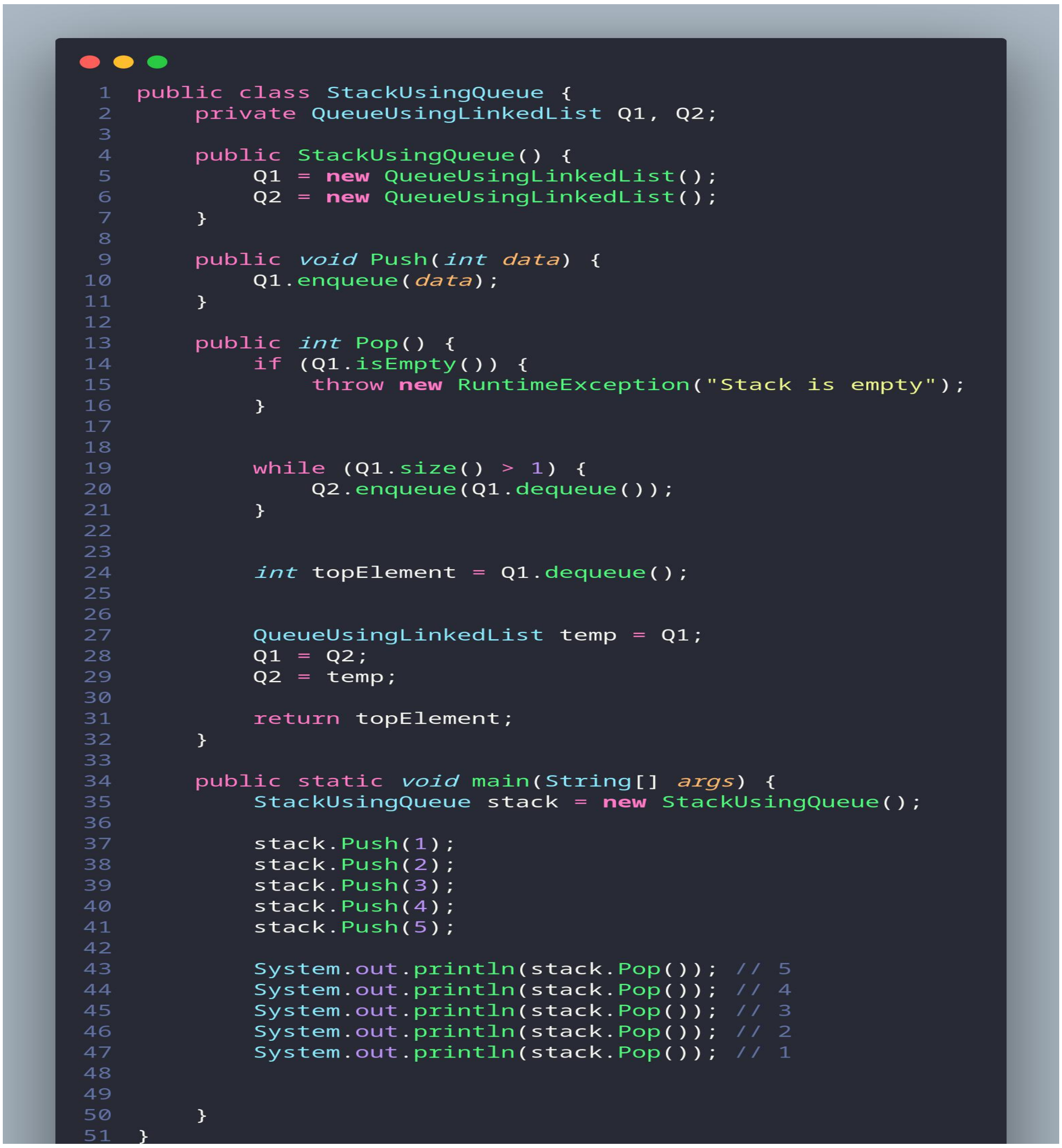
```java
class QueueUsingTwoStacks {
    StackUsingLinkedList s1, s2;

    QueueUsingTwoStacks() {
        s1 = new StackUsingLinkedList();
        s2 = new StackUsingLinkedList();
    }

    public void enqueue(int data) {
        s1.push(data);
    }

    public int dequeue() {
        if (s2.isEmpty()) {
            while (!s1.isEmpty()) {
                s2.push(s1.pop());
            }
        }
        if (s2.isEmpty()) {
            throw new RuntimeException("Queue is empty");
        }
        return s2.pop();
    }

    public static void main(String[] args) {
        int[] keys = {1, 2, 3, 4, 5};
        QueueUsingTwoStacks q = new QueueUsingTwoStacks();

        for (int key : keys) {
            System.out.println("Inserted "+key);
            q.enqueue(key);
        }

        System.out.println("Removed "+q.dequeue());
        System.out.println("Removed "+q.dequeue());
    }
}
```



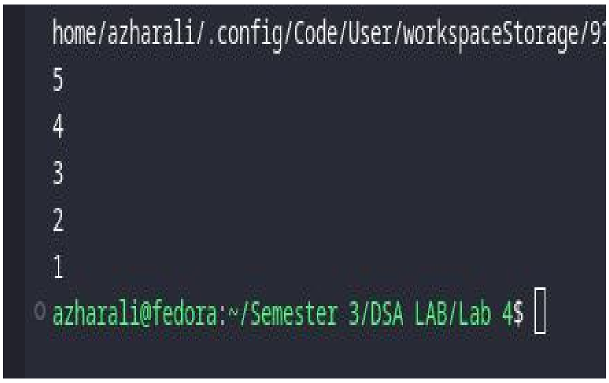
```
onMessages -cp /h
Inserted 1
Inserted 2
Inserted 3
Inserted 4
Inserted 5
Removed 1
Removed 2
azharali@fedora:~
```

df

6. think about the inverse of task 05 (Stack using queue) and implement all the required methods.



```java
public class StackUsingQueue {
    private QueueUsingLinkedList Q1, Q2;

    public StackUsingQueue() {
        Q1 = new QueueUsingLinkedList();
        Q2 = new QueueUsingLinkedList();
    }

    public void Push(int data) {
        Q1.enqueue(data);
    }

    public int Pop() {
        if (Q1.isEmpty()) {
            throw new RuntimeException("Stack is empty");
        }


        while (Q1.size() > 1) {
            Q2.enqueue(Q1.dequeue());
        }


        int topElement = Q1.dequeue();


        QueueUsingLinkedList temp = Q1;
        Q1 = Q2;
        Q2 = temp;

        return topElement;
    }

    public static void main(String[] args) {
        StackUsingQueue stack = new StackUsingQueue();

        stack.Push(1);
        stack.Push(2);
        stack.Push(3);
        stack.Push(4);
        stack.Push(5);

        System.out.println(stack.Pop()); // 5
        System.out.println(stack.Pop()); // 4
        System.out.println(stack.Pop()); // 3
        System.out.println(stack.Pop()); // 2
        System.out.println(stack.Pop()); // 1


    }
}
```



```
home/azharali/.config/Code/User/workspaceStorage/9
5
4
3
2
1
azharali@fedora:~/Semester 3/DSA LAB/Lab 4$
```

df

df