CS162 ASSIGNMENT 6

NAME:

ARCHIT AGRAWAL

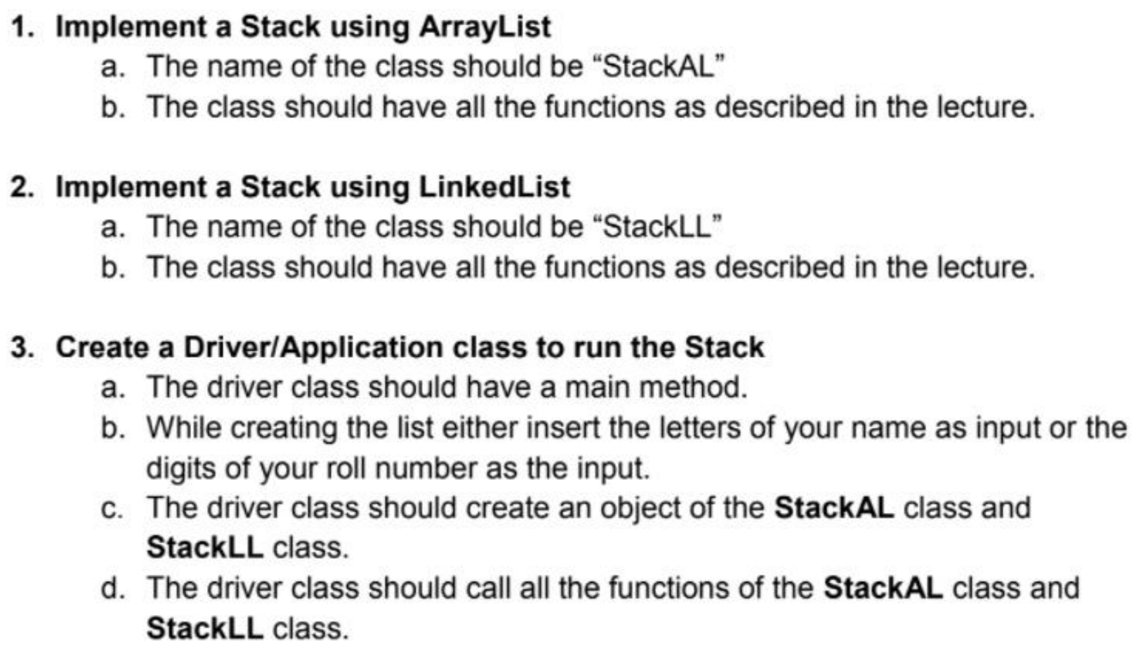
ROLL NO. :

202052307

SECTION:

A

Question



1. **Implement a Stack using ArrayList**
    a. The name of the class should be "StackAL"
    b. The class should have all the functions as described in the lecture.

2. **Implement a Stack using LinkedList**
    a. The name of the class should be "StackLL"
    b. The class should have all the functions as described in the lecture.

3. **Create a Driver/Application class to run the Stack**
    a. The driver class should have a main method.
    b. While creating the list either insert the letters of your name as input or the digits of your roll number as the input.
    c. The driver class should create an object of the **StackAL** class and **StackLL** class.
    d. The driver class should call all the functions of the **StackAL** class and **StackLL** class.

CODE

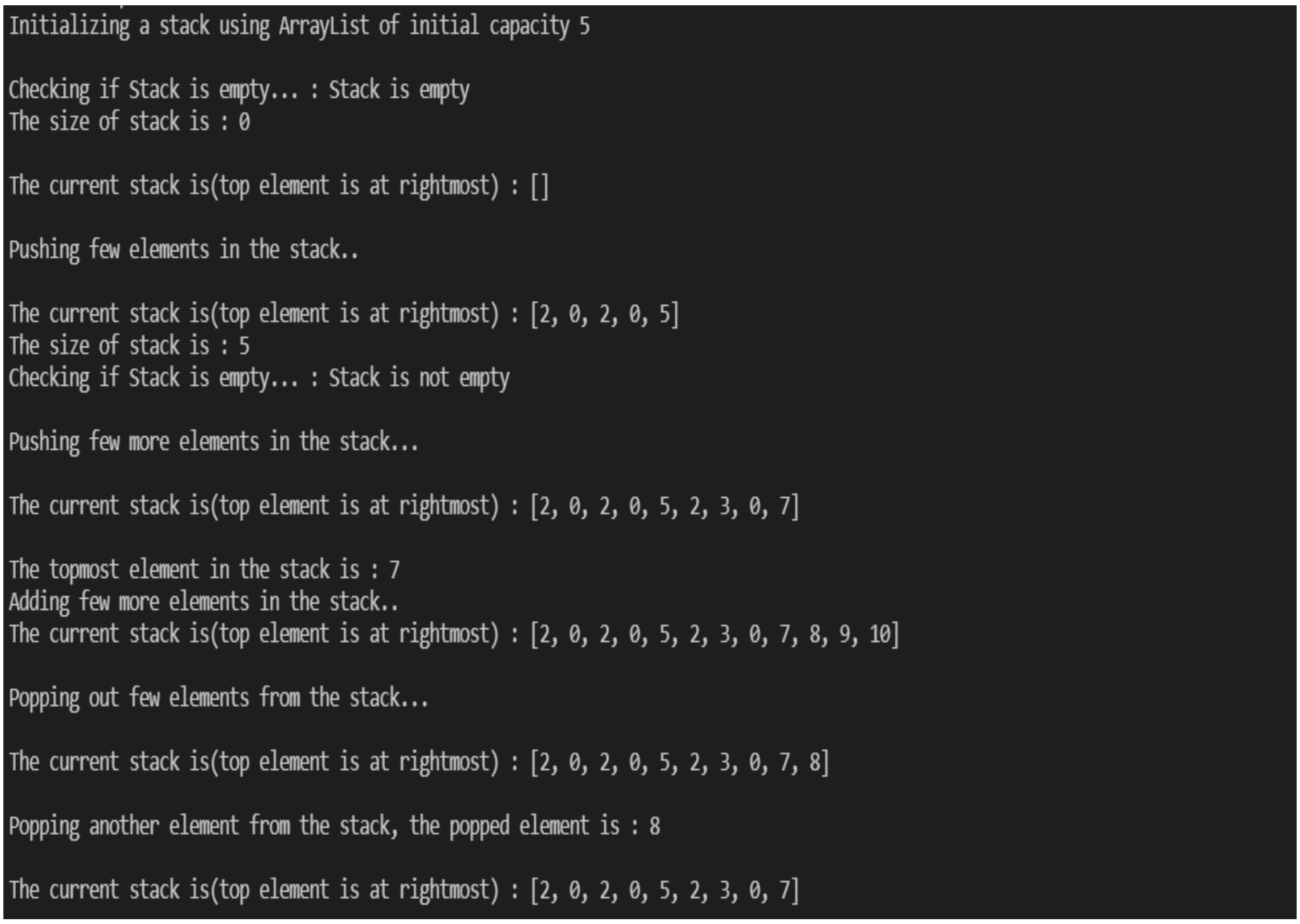
```java
class ArrayList <T>{

    protected T[] arr;
    protected int size;

    public ArrayList(int initialCapacity){
        if(initialCapacity < 1){
            throw new IllegalArgumentException("Initial Capacity must be >= 1"
);
        }
        arr = (T[])new Object[initialCapacity];
        size = 0;
    }

    public ArrayList(){
        this(10);
    }

    public void addToFront(T obj){
        add(obj, 0);
    }

    public void addToRear(T obj){
        add(obj, size);
    }

    public boolean isEmpty(){
        return size == 0;
    }

    public int size(){
        return size;
    }

    public T removeFront(){
        if(size == 0){
            throw new IllegalArgumentException("Array List is Empty");
        }
        T obj = remove(0);
        return obj;
    }
    public T removeRear(){
        if(size == 0){
            throw new IllegalArgumentException("Array List is Empty");
        }
        T obj = remove(size - 1);
        return obj;
    }

    public void checkIndex(int index){
        if(index < 0 || index > size - 1){
            throw new IndexOutOfBoundsException("Index = "+ index + " Size = "
+size);
        }
    }

    public T get(int index){
        checkIndex(index);
        return arr[index];
    }

    public int indexOf(T obj){
        for(int i = 0; i < size; i++){
            //the 'if' condition used is just to make sure that there will be
no 'NullPointerException'
            if((arr[i] == null && obj == null) || (arr[i] != null && arr[i].eq
uals(obj))){
                return i;
            }
        }
        return -1;
    }

    public T remove(int index){
        checkIndex(index);
        T temp = arr[index];
        for(int i = index + 1; i < size; i++){
            arr[i - 1] = arr[i];
        }
        size--;
        arr[size] = null;
        return temp;
    }

    public void changeSize(){
        T []newArray = (T[])new Object[2 * size];
        int i = 0;
        for(T o : arr){
            newArray[i++] = o;
        }
        this.arr = newArray;
    }

    protected void add(T obj, int index){
        if(index < 0 || index > size){
            throw new IndexOutOfBoundsException("Index = "+ index +" Size = "+
size);
        }
        //checking capacity
        if(size == arr.length){
            changeSize();
        }
        //shift elements to right
        for(int i = size; i > index; i--){
            arr[i] = arr[i - 1];
        }
        arr[index] = obj;
        size++;
    }

    public String toString(){

        StringBuilder sb = new StringBuilder();
        sb.append("[");
        for(int i = 0; i < size; i++){
            sb.append((arr[i] != null ? arr[i].toString() : "null"));
            if(i < size - 1){
                sb.append(", ");
            }
        }
        sb.append("]");
        return sb.toString();
    }

}

class StackAL<T> extends ArrayList<T> {
    StackAL(int initialCapacity){
        super(initialCapacity);
    }


    public void push(T data){
        super.add(data, size);
    }
    public T pop(){
        if(super.size() == 0){
            throw new IllegalArgumentException("Nothing to pop, Stack is empty
.");
        }
        return super.remove(size - 1);
    }

    public T peek(){
        if(super.isEmpty()){
            throw new IllegalArgumentException("Stack is empty.");
        }
        return super.get(size - 1);
    }

    public int search(T data){
        int i = super.indexOf(data);
        return i == -1 ? -1 : (size - 1);
    }

    public String display(){
        return super.toString();
    }

    public void addToFront(T obj){}

    public void addToRear(T obj){}

    public T remove(int index){
        return null;
    }

    public T removeFront(){
        return null;
    }

    public T removeRear(){
        return null;
    }

    public T get(int index){
        return null;
    }

    public void checkIndex(int index){}
    public String toString(){
        return null;
    }
}

class Node<T>{
    T data;
    Node next;

    public Node(){}

    public Node(T element){
        this.data = element;
    }

    public Node(T element, Node addr){
        this.data = element;
        this.next = addr;
    }
}

class LinkedList<T>{

    protected Node<T> head;
    protected int size;

    public LinkedList(){
        head = null;
        size = 0;
    }

    //The method append adds an element at the end (index = size) of the linke
d list.
    public void append(T data){
        Node<T> newNode = new Node<T>(data);
        if (head == null) {
            head = newNode;
        } else {
            Node <T> lastNode = head;
            while (lastNode.next != null) {
                lastNode = lastNode.next;
            }
            lastNode.next = newNode;
        }
        size++;
    }
    //The method add adds a given element at the given index.
    public void add(int index, T data){

        if(index < 0 || index > size){
            throw new IndexOutOfBoundsException("Index = "+index +" Size = "+s
ize);
        }

        if(index == 0){
            Node<T> newNode = new Node<T>(data);
            newNode.next = head;
            head = newNode;
        } else {
            Node<T> newNode = new Node<T>(data);
            Node<T> prevNode = head;
            int i = 0;
            while(i < index - 1){
                prevNode = prevNode.next;
                i++;
            }
            newNode.next = prevNode.next;
            prevNode.next = newNode;
        }
        size++;
    }

    //The method removeKey removes the first occurence of the element passed t
o it from the linked list.
    public void removeKey(T key){
        if(size == 0){
            return;
        }
        if(head.data == key){
            head = head.next;

        } else {
            Node<T> currNode = head.next;
            Node<T> prevNode = head;

            while(currNode != null){
                //to prevent NullPointerException
                if(currNode.data.equals(key)){
                    prevNode.next = currNode.next;
                    break;
                } else {
                    prevNode = currNode;
                    currNode = currNode.next;
```

```java
                }
            }
        }
        size--;
    }

    //The method removeIndex removes the element from the index passed to it a
nd returns the removed element.
    public T removeIndex(int index){
        checkIndex(index);
        T removed;

        if(index == 0){
            removed = head.data;
            head = head.next;
        } else {
            Node<T> currNode = head.next;
            Node<T> prevNode = head;
            int i = 0;
            while(i < index - 1){
                prevNode = prevNode.next;
                currNode = currNode.next;
                i++;
            }
            removed = currNode.data;
            prevNode.next = currNode.next;
        }
        size--;
        return removed;
    }

    //This method prints the linked list in formatted way
    public void printLinkedList() {
        Node<T> currentNode = head;
        while(currentNode != null){
            System.out.print(currentNode.data + " -> ");
            currentNode = currentNode.next;
        }
        System.out.println();
    }

    public boolean isEmpty(){
        return size == 0;
    }

    public int size(){
        return size;
    }

    public void checkIndex(int index){
        if(index < 0 || index >= size){
            throw new IndexOutOfBoundsException("Index = "+index +" size = "+
size);
        }
    }

    public T get(int index){
        checkIndex(index);
        Node<T> currNode = head;
        int i = 0;
        while(i != index){
            currNode = currNode.next;
            i++;
        }
        return currNode.data;
    }

    public int indexOf(T data){
        Node<T> temp = head;
        int index = 0;
        while(temp != null){
            if(temp.data.equals(data)){
                return index;
            }
            index++;
            temp = temp.next;
        }
        return -1;
    }

}

class StackLL<T> extends LinkedList<T>{

    public StackLL(){
        super();
    }

    public void push(T data){
        super.append(data);
    }

    public T pop(){
        if(super.isEmpty()){
            throw new IllegalArgumentException("Nothing to pop, Stack is empty
.");
        }
        return super.removeIndex(size - 1);
    }

    public T peek(){
        if(super.isEmpty()){
            throw new IllegalArgumentException("Stack is empty.");
        }
        return super.get(size - 1);
    }

    public int search(T data){
        int i = super.indexOf(data);
        return i == -1 ? -1 : (size - 1);
    }

    public void display(){
        super.printLinkedList();
    }

    public void append(T data){}

    public void add(int index, T data){}

    public void removeKey(T key){}

    public T removeIndex(int index){
        return null;
    }

    public void printLinkedList(){}

    public void checkIndex(int index){}

    public T get(int index){
        return null;
    }

}

public class Driver{
    public static void main(String []args){

        System.out.println("Initializing a stack using ArrayList of initial ca
pacity 5");
```

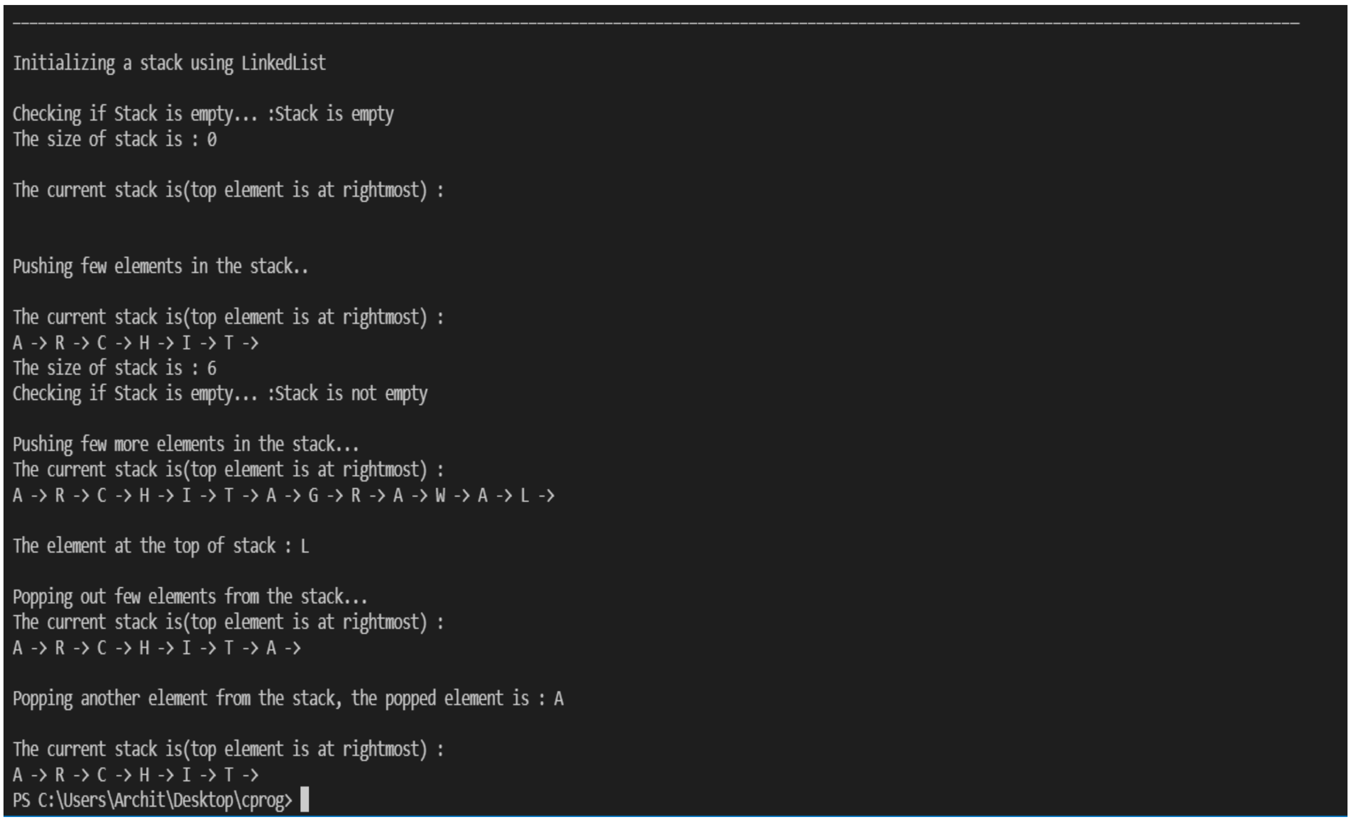
```
        StackAL <Integer> stackALObject = new StackAL<Integer>(5);
        System.out.println();

        System.out.print("Checking if Stack is empty... : ");
        if(stackALObject.isEmpty()){
            System.out.println("Stack is empty");
        } else {
            System.out.println("Stack is not empty");
        }

        System.out.println("The size of stack is : "+stackALObject.size());
        System.out.println();

        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());

        System.out.println();
        System.out.println("Pushing few elements in the stack..");
        stackALObject.push(2);
        stackALObject.push(0);
        stackALObject.push(2);
        stackALObject.push(0);
        stackALObject.push(5);

        System.out.println();
        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());
        System.out.println("The size of stack is : "+stackALObject.size());
        System.out.print("Checking if Stack is empty... : ");
        if(stackALObject.isEmpty()){
            System.out.println("Stack is empty");
        } else {
            System.out.println("Stack is not empty");
        }
        System.out.println();

        System.out.println("Pushing few more elements in the stack...");
        stackALObject.push(2);
        stackALObject.push(3);
        stackALObject.push(0);
        stackALObject.push(7);
        System.out.println();
        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());
        System.out.println();
        System.out.println("The topmost element in the stack is : "+stackALObj
ect.peek());
        System.out.println("Adding few more elements in the stack..");

        stackALObject.push(8);
        stackALObject.push(9);
        stackALObject.push(10);
        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());
        System.out.println();
        System.out.println("Popping out few elements from the stack...");
        stackALObject.pop();
        stackALObject.pop();
        System.out.println();

        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());
        System.out.println();
        System.out.println("Popping another element from the stack, the popped
 element is : "+stackALObject.pop());
        System.out.println();
        System.out.println("The current stack is(top element is at rightmost)
: "+stackALObject.display());

        System.out.println("__________________________________________________
______________________________________________________________________________
________________________");
        System.out.println();
        System.out.println("Initializing a stack using LinkedList");
        StackLL <Character> stackLLObject = new StackLL<Character>();
        System.out.println();

        System.out.print("Checking if Stack is empty... :");
        if(stackLLObject.isEmpty()){
            System.out.println("Stack is empty");
        } else {
            System.out.println("Stack is not empty");
        }

        System.out.println("The size of stack is : "+stackLLObject.size());
        System.out.println();

        System.out.println("The current stack is(top element is at rightmost)
: ");
        stackLLObject.display();

        System.out.println();
        System.out.println("Pushing few elements in the stack..");
        stackLLObject.push('A');
        stackLLObject.push('R');
        stackLLObject.push('C');
        stackLLObject.push('H');
        stackLLObject.push('I');
        stackLLObject.push('T');

        System.out.println();
        System.out.println("The current stack is(top element is at rightmost)
: ");
        stackLLObject.display();
        System.out.println("The size of stack is : "+stackLLObject.size());
        System.out.print("Checking if Stack is empty... :");
        if(stackLLObject.isEmpty()){
            System.out.println("Stack is empty");
        } else {
            System.out.println("Stack is not empty");
        }
        System.out.println();

        System.out.println("Pushing few more elements in the stack...");
        stackLLObject.push('A');
        stackLLObject.push('G');
        stackLLObject.push('R');
        stackLLObject.push('A');
        stackLLObject.push('W');
        stackLLObject.push('A');
        stackLLObject.push('L');

        System.out.println("The current stack is(top element is at rightmost)
: ");
        stackLLObject.display();
        System.out.println();

        System.out.println("The element at the top of stack : "+stackLLObject.
peek());

        System.out.println();
        System.out.println("Popping out few elements from the stack...");

        for(int i = 0; i < 6; i++){
            stackLLObject.pop();
        }

        System.out.println("The current stack is(top element is at rightmost)
: ");
```

```
        stackLLObject.display();
        System.out.println();

        System.out.println("Popping another element from the stack, the popped
 element is : "+stackLLObject.pop());
        System.out.println();
        System.out.println("The current stack is(top element is at rightmost)
: ");
        stackLLObject.display();

    }
}
```

OUTPUT



```
Initializing a stack using ArrayList of initial capacity 5

Checking if Stack is empty... : Stack is empty
The size of stack is : 0

The current stack is(top element is at rightmost) : []

Pushing few elements in the stack..

The current stack is(top element is at rightmost) : [2, 0, 2, 0, 5]
The size of stack is : 5
Checking if Stack is empty... : Stack is not empty

Pushing few more elements in the stack...

The current stack is(top element is at rightmost) : [2, 0, 2, 0, 5, 2, 3, 0, 7]

The topmost element in the stack is : 7
Adding few more elements in the stack..
The current stack is(top element is at rightmost) : [2, 0, 2, 0, 5, 2, 3, 0, 7, 8, 9, 10]

Popping out few elements from the stack...

The current stack is(top element is at rightmost) : [2, 0, 2, 0, 5, 2, 3, 0, 7, 8]

Popping another element from the stack, the popped element is : 8

The current stack is(top element is at rightmost) : [2, 0, 2, 0, 5, 2, 3, 0, 7]
```



```
------------------------------------------------------------------------------------------------------------------------

Initializing a stack using LinkedList

Checking if Stack is empty... :Stack is empty
The size of stack is : 0

The current stack is(top element is at rightmost) :


Pushing few elements in the stack..

The current stack is(top element is at rightmost) :
A -> R -> C -> H -> I -> T ->
The size of stack is : 6
Checking if Stack is empty... :Stack is not empty

Pushing few more elements in the stack...
The current stack is(top element is at rightmost) :
A -> R -> C -> H -> I -> T -> A -> G -> R -> A -> W -> A -> L ->

The element at the top of stack : L

Popping out few elements from the stack...
The current stack is(top element is at rightmost) :
A -> R -> C -> H -> I -> T -> A ->

Popping another element from the stack, the popped element is : A

The current stack is(top element is at rightmost) :
A -> R -> C -> H -> I -> T ->
PS C:\Users\Archit\Desktop\cprog>
```

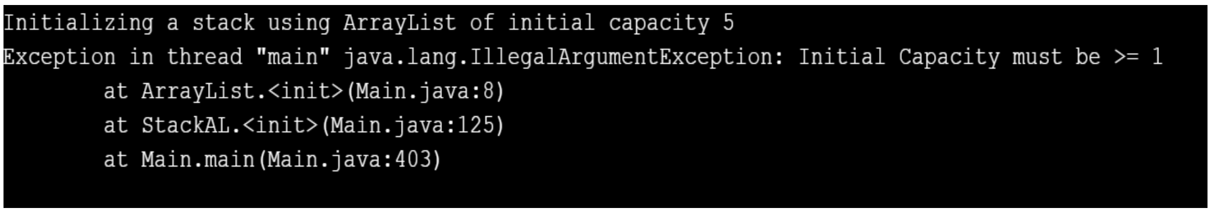
Code for throwing Exceptions -:

The changes required in the code above in order to throw exceptions are to be done in Driver class only.

Hence, instead of pasting the whole code again and again, only the class Driver code is pasted.

- *For throwing IllegalArgumentException in initializing Stack using ArrayList*

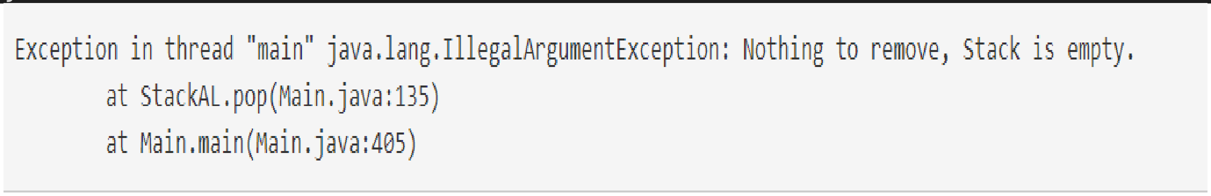
```
public class Driver{
    public static void main(String []args){

        System.out.println("Initializing a stack using ArrayList of initial ca
pacity 5");
        StackAL <Integer> stackALObject = new StackAL<Integer>(-1);
        System.out.println();
    }
}
```



```
Initializing a stack using ArrayList of initial capacity 5
Exception in thread "main" java.lang.IllegalArgumentException: Initial Capacity must be >= 1
        at ArrayList.<init>(Main.java:8)
        at StackAL.<init>(Main.java:125)
        at Main.main(Main.java:403)
```

- ***<u>For throwing IllegalArgumentException when pop method is called on empty Stack using ArrayList</u>***

```
public class Main{
    public static void main(String []args){

        System.out.println("Initializing a stack using ArrayList of initial ca
pacity 5");
        StackAL <Integer> stackALObject = new StackAL<Integer>(5);

        stackALObject.pop();
    }
}
```



```
Exception in thread "main" java.lang.IllegalArgumentException: Nothing to remove, Stack is empty.
        at StackAL.pop(Main.java:135)
        at Main.main(Main.java:405)
```

- ***<u>For throwing IllegalArgumentException when peek method is called on empty Stack using ArrayList</u>***

```
public class Main{
    public static void main(String []args){

        System.out.println("Initializing a stack using ArrayList of initial ca
pacity 5");
        StackAL <Integer> stackALObject = new StackAL<Integer>(5);

        System.out.println(stackALObject.peek());
    }
}
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Stack is empty.
        at StackAL.peek(Main.java:142)
        at Main.main(Main.java:405)
```

- ***For throwing IllegalArgumentException when pop method is called on empty Stack using LinkedList***

```
public class Main{
    public static void main(String []args){
        System.out.println("Initializing a stack using LinkedList");
        StackLL <Character> stackLLObject = new StackLL<Character>();
        System.out.println();

        stackLLObject.pop();
    }
}
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Nothing to pop, Stack is empty.
        at StackLL.pop(Main.java:361)
        at Main.main(Main.java:467)
```

- ***For throwing IllegalArgumentException when peek method is called on empty Stack using LinkedList***

```
public class Main{
    public static void main(String []args){
        System.out.println("Initializing a stack using LinkedList");
        StackLL <Character> stackLLObject = new StackLL<Character>();
        System.out.println();

        stackLLObject.peek();
    }
}
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Stack is empty.
        at StackLL.peek(Main.java:368)
        at Main.main(Main.java:467)
```