ASSIGNMENT

THIRD SEMESTER

Analyzing Data Structure and Algorithms

By:

Pranish Raj Tuladhar (C30109220142)

BACHELOR OF INFORMATION & COMMUNICATION TECHNOLOGY

SCHOOL OF SCIENCE & TECHNOLOGY

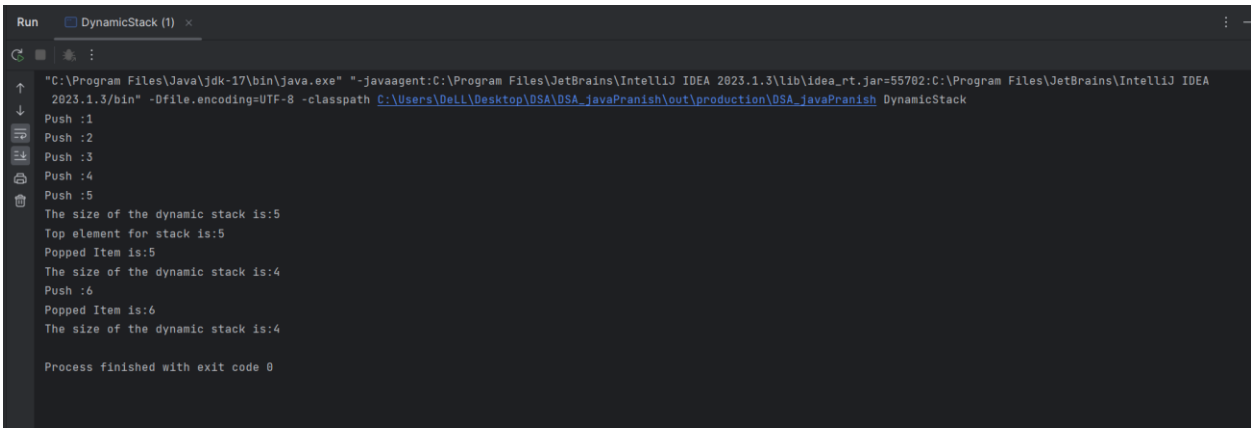
VIRINCHI COLLEGE

Write a Java program to implement a dynamic stack, that automatically adjusts its size as elements are added or removed.

```
import java.util.LinkedList;
import java.util.List;

public class DynamicStack {
    private List<Integer> stackDynamic = new LinkedList<>();

    public void push(int item) {
        stackDynamic.add(item);
        System.out.println("Push :" + item);
    }

    public void pop() {
        if (stackDynamic.isEmpty()) {
            System.out.println("Stack Underflow");
            return;
        }
        int lastIndex = stackDynamic.size() - 1;
        int poppedItem = stackDynamic.get(lastIndex);
        stackDynamic.remove(lastIndex);
        System.out.println("Popped Item is:" + poppedItem);
    }

    public void peek() {
        if (stackDynamic.isEmpty()) {
            System.out.println("Stack Underflow");
            return;
        }
        System.out.println("Top element for stack is:" +
stackDynamic.get(stackDynamic.size() - 1));
    }

    public void size() {
        System.out.println("The size of the dynamic stack is:" +
stackDynamic.size());
    }

    public static void main(String[] args) {
        DynamicStack ds = new DynamicStack();

        ds.push(1);
        ds.push(2);
        ds.push(3);
        ds.push(4);
        ds.push(5);

        ds.size();
        ds.peek();

        ds.pop();

        ds.size();
```

```
        ds.push(6);
        ds.pop();
        ds.size();
    }
}
```

SCREEN SHOT



2) Develop a Java program demonstrating the implementation of a linear queue:
This program should prompt the user to specify the desired queue size and user needs to input the enqueue elements accordingly. Also, show dequeue operations.

```
import java.util.Scanner;
import java.util.EmptyStackException;

class LinearQueue {
    private int front, rear, capacity;
    private int[] queue;

    public LinearQueue(int capacity) {
        this.capacity = capacity;
        queue = new int[capacity];
        front = rear = -1;
    }

    public boolean isEmpty() {
        return front == -1;
    }

    public boolean isFull() {
        return rear == capacity - 1;
    }

    public void enqueue(int item) {
        if (isFull()) {
            System.out.println("Queue is full");
            return;
        }
```

```java
        if (isEmpty())
            front = 0;
        queue[++rear] = item;
    }

    public int dequeue() {
        if (isEmpty()) {
            throw new EmptyStackException();
        }
        int item = queue[front];
        if (front == rear)
            front = rear = -1;
        else
            front++;
        return item;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the capacity of the queue: ");
        int capacity = scanner.nextInt();
        LinearQueue queue = new LinearQueue(capacity);

        System.out.println("Linear Queue Operations:");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Exit");

        int choice;
        do {
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    if (queue.isFull()) {
                        System.out.println("Queue is full");
                    } else {
                        for(int i = 0; capacity > i ; i++) {
                            System.out.print("Enter element to enqueue: ");
                            int element = scanner.nextInt();
                            queue.enqueue(element);
                        }
                    }
                    break;
                case 2:
                    if (queue.isEmpty()) {
                        System.out.println("Queue is empty");
                    } else {
                        int dequeued = queue.dequeue();
                        System.out.println("Dequeued element: " + dequeued);
                    }
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice");
```

```
            }
        } while (choice != 3);

        scanner.close();
    }

}
```

SCREEN SHOT