Q1. Explain Stack Data Structure ?
Ans: A stack is a linear data structure that store data inform of FIrst In Last Out Order / Last IN First Out. Data of Stack insertion and deletion perform only one end (top)

Application Of Stack Data Structure

1. Function calling
2. Tower of Hanoi
3. Undo and Redo

Operations of Stack

1. PUSH(): Adds data into the Stack Data Structure
2. POP(): Delete data from the stack(top element is removed and return)
3. Peek(): Return top element of the stack without removal
4. isEmpty(): To check the stack is Empty or not
5. isFull():To check stack is full or not

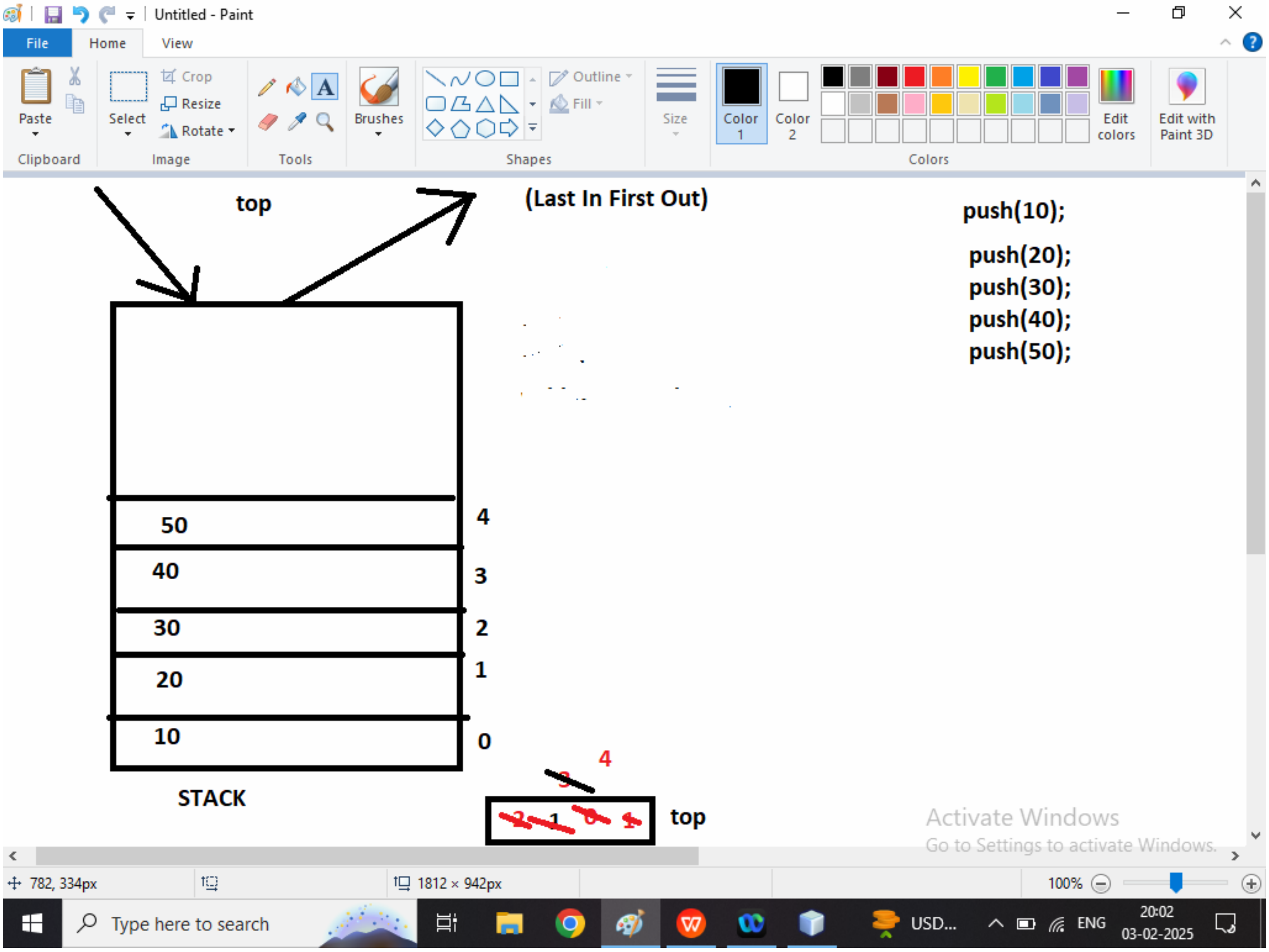
Implementation of stack:

1. Array
2. Linked List
3. Queue

Time Compexity of All Operations of the Stack:

Stack Implementation Using Array
Push Operation Algo
Step1: To Check Over Flow Condition

```
if(top==S.length-1){
//stack overflow
}
Step2: Increase Top Variable By 1
top++;//top=0
//step3: Assign Value in Array
S[top]=data;
```

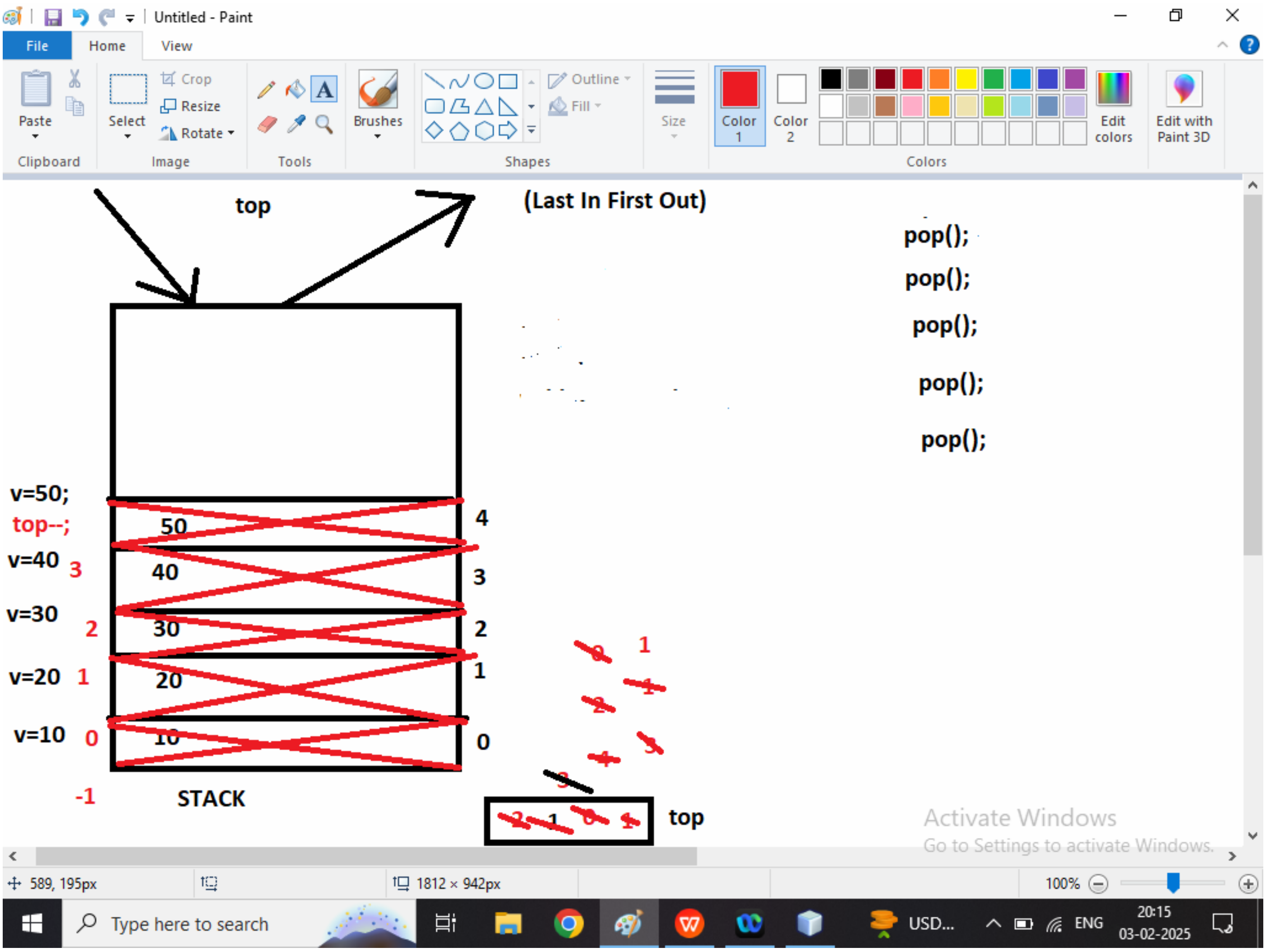


```
push(10);
push(20);
push(30);
push(40);
push(50);
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

/**
 *
 * @author Admin
 */
public class StackDemo {
 static   int STACK[]=new int[5];
   static int top=-1;
   //Member data

   public static boolean isEmpty(){
      return top==-1;
   }
   public static boolean isFull(){
      return top==STACK.length-1;
   }
   public static void push(int data){
      //step1: To check Over flow
      if(isFull()){
          System.out.println("Stack Over flow");
      }
      //step2: Increase Top by 1
      top++;
      //step3: Assign value into the stack
      STACK[top]=data;
      System.out.println("Data Insert into the stack is success");
   }


   public static void display(){
      if(isEmpty()){
          System.out.println("Stack is Empty");
      }else{
          System.out.println("Print Data of the Stack ");
          for(int i=top;i>=0;i--){
              System.out.println("===>"+STACK[i]);
          }
      }
   }

   public static void main(String[] args) {
      push(10);
      push(20);
      push(30);
      push(40);
      push(50);
      //push(60);
      display();
   }
}
```

Algo of POP() Operation:

```
Step1: Check Under Flow
If(isEmpty()){
Sop(“Stack Is Under flow”);
}

//step2:Store top element data into anoter variable
Int v=STACK[top];
//step3: Decrease top by 1
top--;
//step4: Return the deleted element
Return v;
```



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

/**
 *
 * @author Admin
 */
public class StackDemo {

    static int STACK[] = new int[5];
    static int top = -1;
    //Member data

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

    public static boolean isFull() {
        return top == STACK.length - 1;
    }

    public static void push(int data) {
        //step1: To check Over flow
        if (isFull()) {
            System.out.println("Stack Over flow");
        }
        //step2: Increase Top by 1
        top++;
        //step3: Assign value into the stack
        STACK[top] = data;
        System.out.println("Data Insert into the stack is success");
    }

    public static int pop() {
        //step1: Check Under flow
        int r = -1;
        if (isEmpty()) {
            System.out.println("Under Flow");
        } else {
            //step2: Assign data into another variable
            r = STACK[top];
            //step3: decreace top by 1
            top--;
        }
        //step4: Return value of v
        return r;
    }

    public static void display() {
        if (isEmpty()) {
            System.out.println("Stack is Empty");
        } else {
            System.out.println("Print Data of the Stack ");
            for (int i = top; i >= 0; i--) {
                System.out.println("===>" + STACK[i]);
            }
        }
    }

    public static int peek(){
        if(!isEmpty()){
            return STACK[top];
        }else{
            return -1;
        }
    }
    public static void main(String[] args) {
        push(10);
        push(20);
        push(30);
        push(40);
        push(50);
        //push(60);
        display();
        System.out.println("Deleted Element : " + pop());
        System.out.println("Deleted Element : " + pop());
        System.out.println("Top Element of the Stack : "+peek());
        System.out.println("Deleted Element : " + pop());
        System.out.println("Deleted Element : " + pop());
        System.out.println("Deleted Element : " + pop());
         System.out.println("Deleted Element : "+pop());
    }
}
```

Q2. Explain Queue data Structure?
Ans: if we want to store data inform of FIRST IN FIRST OUT Order then we should go for Queue Data Structure

Operations of Queue Data Structure

1. Enque():Insert data into the Queue from the rear end(increase rear by 1)
2. Deque(): delete data from the queue using Front end(Increase front by 1)
3. isEmpty(): To check queue is Empty or not
4. isFull(): To Check Queue is FUll or not
5. Peek(): Return Return Front Element of the Queue
6. Display():print all data of Queue

Algorithm of Enque operation

```
Step1: First Check Over flow Condition
Step2: To check Queue is Empty Increase front and rear by 1
Step3: If queue is Not Empty then increase rear by 1
Step4: Insert data into queue
Q[rear]=data;
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

public class QueueDEMO {

    static int Q[] = new int[5];
    static int front = -1;
    static int rear = -1;

    public static boolean  isEmpty(){
        return front==-1 && rear==-1;
    }
    public static boolean isFull(){
        return rear==Q.length-1;
    }
    public int enq(int data) {
        // Step1: First Check Over flow Condition
        //Step2: To check Queue is Empty Increase front and rear by 1
        //Step3: If queue is Not Empty then increase rear by 1
        //Step4: Insert data into queue
    }

}
```