Solution A

What's the biggest number on the stack?

Problem Statement

Manipulate a stack according to the given push and pop commands and then output the biggest number that is left on the stack. If a pop command is issued for an empty stack then nothing should happen.

Input Format

The first line is a number N, which indicates the number of commands to follow.

This is followed by N lines, each of which consists of the word PUSH or POP. The word PUSH will be followed by an integer n.

Output Format

Output the biggest integer that is left anywhere on the stack following the commands. If the stack is empty output "empty".

Constraints

1<=N<=10

-10000<=n<=10000

Sample Input

```
5

PUSH 4

PUSH 8

POP

PUSH 6
```

PUSH 2

Sample Output

6

```
import java.util.Scanner;

public class Solution {
    public static void main(String args[] ) throws Exception {
      Scanner myscanner = new Scanner(System.in);
      Stack mystack = new Stack(10);
         int num = Integer.parseInt(myscanner.nextLine());
         for(int i=0;i<num;i++){
             String command = myscanner.nextLine();
             if(command.equals("POP")){
                 if(!mystack.isEmpty()){
                     mystack.pop();
                 }
             }else{

mystack.push(Integer.parseInt(command.substring(5,command.length()))
);
             }
         }
         if(!mystack.isEmpty()){
             int record = mystack.pop();
             while(!mystack.isEmpty()){
                 if(mystack.peek()>record){
                     record=mystack.peek();
                 }
                 mystack.pop();
             }
             System.out.println(record);

         }else{
             System.out.println("empty");
         }
    }
}


class Stack{

    private int maxSize;         // size of stack array
    private int[] stackArray;
    private int top;             // top of stack

    public Stack(int s) {          // constructor
```

```
        maxSize = s;               // set array size
        stackArray = new int[maxSize]; // create array
        top = -1;                  // no items yet
    }

    public void push(int j) {      // put item on top of stack
         top++;
       stackArray[top] = j;    // increment top, insert item
    }

    public int pop() {             // take item from top of stack
        return stackArray[top--]; //access item, decrement top
    }

    public int peek() {            // peek at top of stack
        return stackArray[top];
    }

    public boolean isEmpty() {     // true if stack is empty
        return (top == -1);
    }

    public boolean isFull() {       // true if stack is full
        return (top == maxSize-1);
    }

    public void makeEmpty() {       // empty stack
         top=-1;
    }
}
```

Solution B

What's on top of the stack?

Problem Statement

Manipulate a stack according to the given push and pop commands and then output the number that is at the top of the stack. If a pop command is issued for an empty stack then nothing should happen.

Input Format

The first line is a number N, which indicates the number of commands to follow.

This is followed by N lines, each of which consists of the word PUSH or POP. The word PUSH will be followed by an integer n.

Output Format

Output the integer that is at the top of the stack following the given commands. If the stack is empty output "empty".

Constraints

1<=N<=10

-10000<=n<=10000

Sample Input

5

PUSH 4

PUSH 8

POP

POP

PUSH 2

Sample Output

2

```
import java.util.*;

public class Solution {
    public static void main(String args[] ) throws Exception {


        Scanner myscanner = new Scanner(System.in);
      Stack mystack = new Stack(10);
        int num = Integer.parseInt(myscanner.nextLine());
        for(int i=0;i<num;i++){
            String command = myscanner.nextLine();
            if(command.equals("POP")){
                if(!mystack.isEmpty()){
                    mystack.pop();
                }
            }else{

mystack.push(Integer.parseInt(command.substring(5,command.length()))
);
            }
        }
        if(!mystack.isEmpty()){
            System.out.println(mystack.peek());
        }else{
            System.out.println("empty");
        }
    }
}



class Stack{

    private int maxSize;        // size of stack array
    private int[] stackArray;
    private int top;            // top of stack

    public Stack(int s) {          // constructor
        maxSize = s;              // set array size
        stackArray = new int[maxSize]; // create array
        top = -1;                 // no items yet
    }

    public void push(int j) {    // put item on top of stack
         top++;
       stackArray[top] = j;   // increment top, insert item
    }

    public int pop() {            // take item from top of stack
        return stackArray[top--]; //access item, decrement top
    }

    public int peek() {           // peek at top of stack
        return stackArray[top];
    }

    public boolean isEmpty() {   // true if stack is empty
        return (top == -1);
    }

    public boolean isFull() {     // true if stack is full
        return (top == maxSize-1);
    }

    public void makeEmpty() {     // empty stack
         top=-1;
    }
}
```