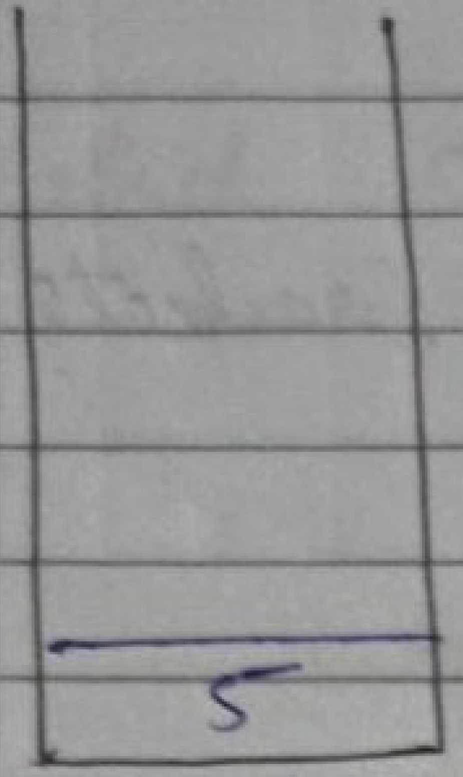
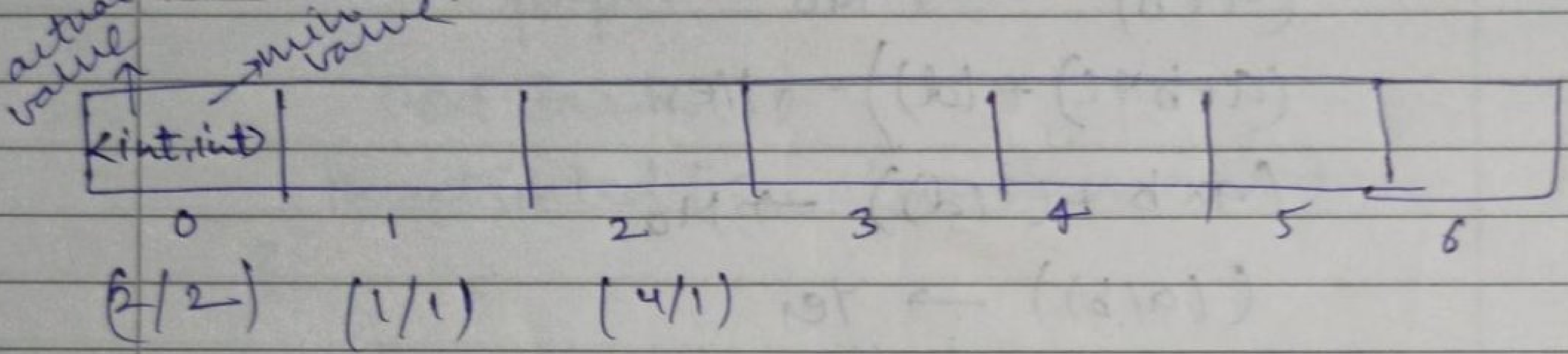
Stack

1) Min stack - (Leetcode)



push(5) ✓ constant time
pop() ✓ constant time

We have to find minimum in t.c. O(1).



Insert karte wqt check karo minimum

code

```
vector<pair<int,int>>st;
 MinStack(){

 }
 void push(int val){
       if(st.empty(){
    pair<int,int> p = make_pair(val,val);
      st.pushback(p);
   }
   else
       pair<int>,int>p;
       p.first = val;
       p.second = min(val, st.back().second);
        st.pushback(p);
   }
 }
```

```
void pop() {
   st.pop_back();
 }

int top() {
  return st.back().first;
 }

int getMin() {
  return st. back().second;
 }
```

2) Longest Valid Parenthesis
Return length of longest valid parenthesis.

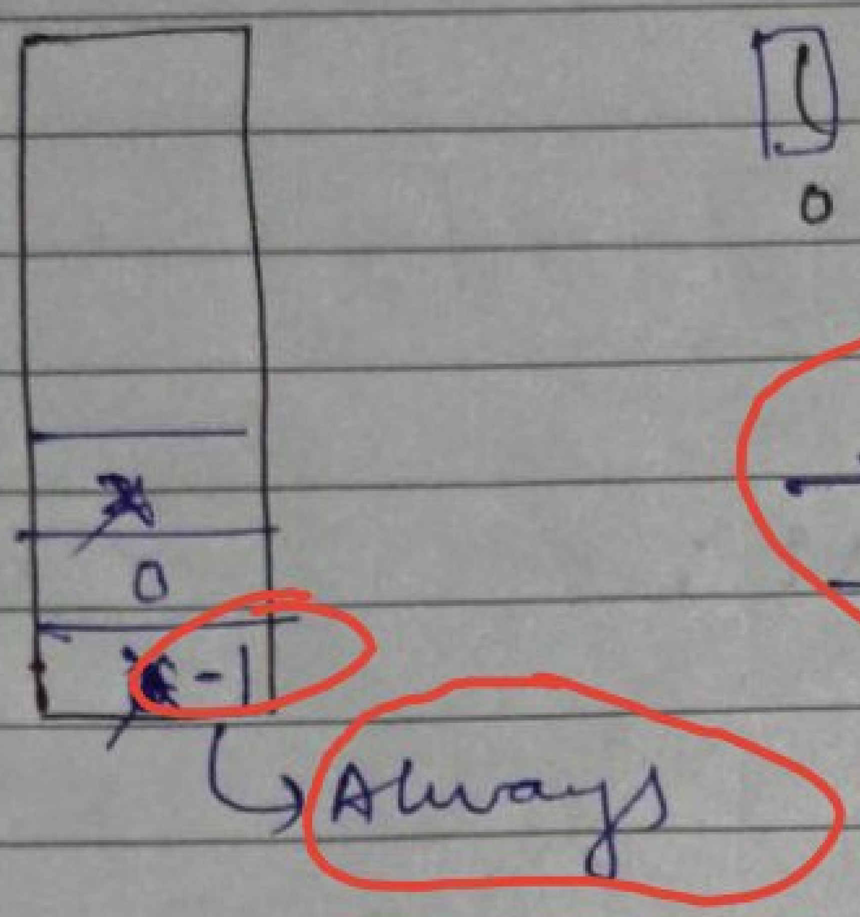
eg. $s = "(\boxed{()}"$
Ans = 2

eg. $")\boxed{()(())})"$
Ans = 4

eg. $)\boxed{()()(())}($
$\downarrow$
Ans = 8

eg. $))))))\boxed{()}$
Ans = 2

Approach-

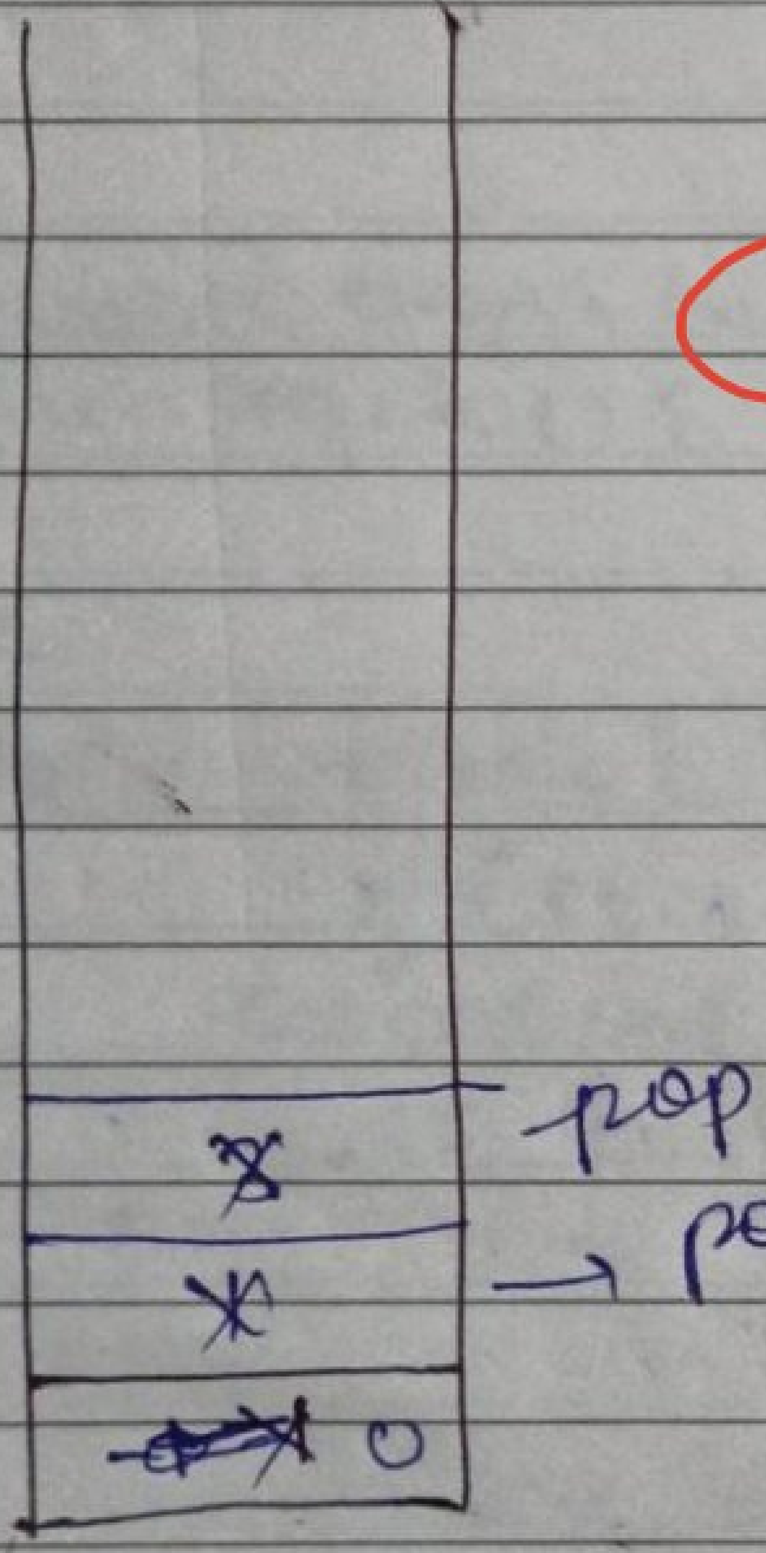


( ) ( )
0 1 2 3

→ (→ insert index.
→) → pop → length
~~ct~~ length = curr - s.top()
1 - (-1)
=> 2

→ (→ insert index
→ pop
length = 3 - (-1)
=> 4

eg



) ( ) ( ) (
0 1 2 3 4 5

) → pop → s.empty()

yes → s.push(i) → curr.index

No → length find

length = 2 - 0
= 2

length = 4 - 0
= 4

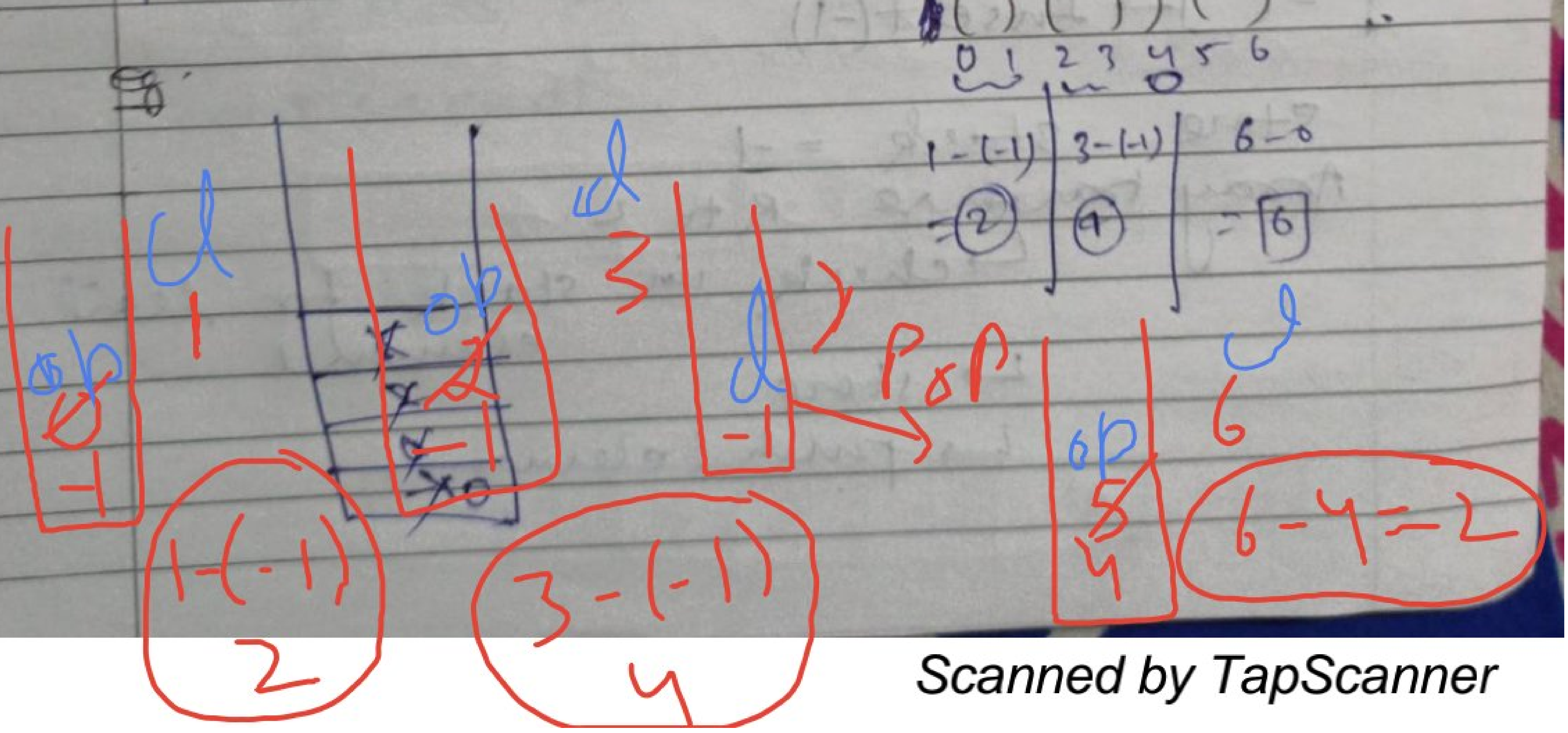
max(2, 4)
= 4 Ans.

Code

```
int longestParenthesis(string s){
  stack<int>st;
   int maxlen=0;
    st.push(-1);
 for(int i=0; i<s.length; i++){
     char ch=s[i];
     if(ch=='('){
            st.push(i);
     }
     else{
         st.pop();
         if(st.empty()){
                 st.push(i);
         }
         else{
           int len = i-st.top();
           maxlen = max(len, maxlen);
         }
      }
  }
  return maxlen;
}
```

Push current index

Eg.



Scanned by TapScanner

Next smaller Element

2	1	4	3

next smaller for 2 is 1

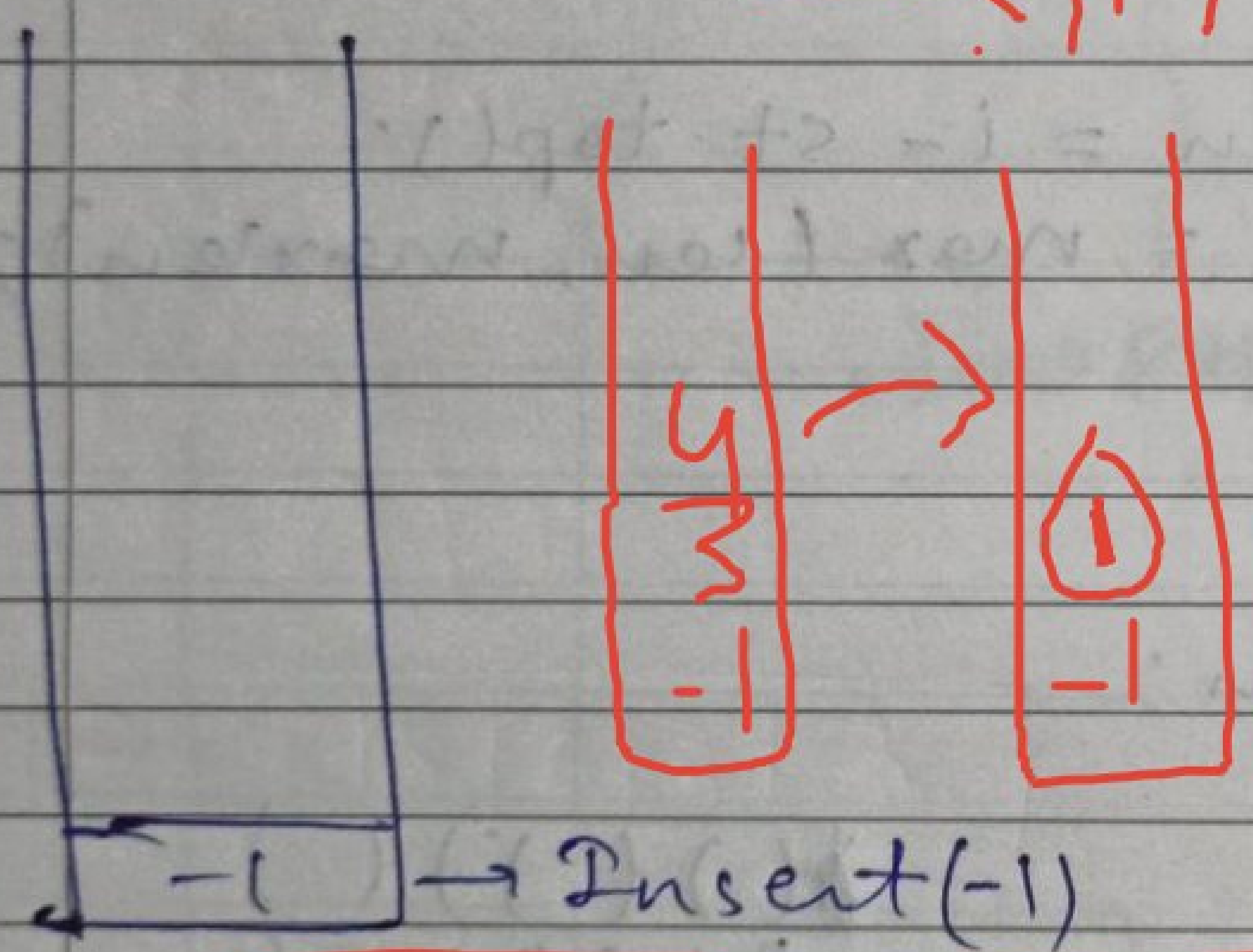
" " for 1 is no element

" " for 4 is 3

" " for 3 is no element

{1 , -1 , 3, -1}

[2, 1, 4, 3]



{1 -1 3 -1}

Store stack = -1
Array traverse R to L
↳ check in stack for least element
↳ store
↳ push (element)