Now the peek element is 3.

At last we have simply printed the elements present within the stack and they are 1 and 3.

Q2. Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.
Open brackets must be closed in the correct order.
Every close bracket has a corresponding open bracket of the same type.

Example 1:
Input: s = "()"
Output: true

Example 2:
Input: s = "()[]{}"
Output: true

Example 3:
Input: s = "(]"
Output: false

Example 4:
Input: s = "({})"
Output: true

Solution :

Code : LP_Code2.java

Output :

```
Enter the string: [[{}]()
The given string is valid ? false
```

Approach :

- We have to keep some data structure which will tell us, whether its opening term comes or not.
- Means suppose we met parentheses '}', but parentheses do not come before already '{', then it can never be valid.
- And the data structure we are going to use is stack.
- And why STACK? Because it will count as the last element that comes in.
- Stack works on the principle of LIFO i.e Last In First Out.
- Let's dry run it in the above example.
- Suppose our string given to us as, s = "(({}))"

Initially, our stack is empty, and looks like, t = [],

s = "(({ }))"
↑
we met an opening bracket, we will say there is a probability of meeting same closing bracket in future, so push into stack,
now stack looks like, t = ['(']
↑ t.top()

s = "(({ }))"
↑
we met an opening bracket, we will say there is a probability of meeting same closing bracket in future, so push into stack,
now stack looks like, t = ['(', '(']
↑ t.top()

s = "(({ }))"
↑
we met an opening bracket, we will say there is a probability of meeting same closing bracket in future, so push into stack,
now stack looks like, t = ['(', '(', '{']
↑ t.top()

s = "(({ }))"
↑
Now, we met an closing bracket, so we say our parentheses will valid only if it encounter the last same opening parentheses of this type, and how we encounter our last opening parentheses, for that we are using stack
t = ['(', '(', '{']
↑ t.top()

so, yes our current bracket is '}' and top of stack is '{', so they are valid,
therefore move forward and pop the peek/top element from stack,
t = ['(', '(',]
↑ t.top()
so, yes our current bracket is ')' and top of stack is '(', so they are valid,
therefore move forward and pop the top from stack,
t = ['(']

s = "(({ }))"
↑
Now, we met an closing bracket, so we say our parentheses will valid only if it encounter the last same opening parentheses of this type, and how we encounter our last opening parentheses, for that we are using stack
our current stack looks like, t = ['(']
↑ t.top()
so, yes our current bracket is ')' and top of stack is '(', so they are valid,
therefore move forward and pop the top from stack,
t = [ ]

Since, we are on the last index and our stack is empty,
therefore it is valid.

If the stack is not found empty then we have concluded that for some opening bracket, there does not exist a valid closing parentheses so the given string is invalid.