## 678. Valid Parenthesis String

Given a string containing only three types of characters: '(', ')' and '*', write a function to check whether this string is valid. We define the validity of a string by these rules:

1. Any left parenthesis `'('` must have a corresponding right parenthesis `')'`.
2. Any right parenthesis `')'` must have a corresponding left parenthesis `'('`.
3. Left parenthesis `'('` must go before the corresponding right parenthesis `')'`.
4. `'*'` could be treated as a single right parenthesis `')'` or a single left parenthesis `'('` or an empty string.
5. An empty string is also valid.

**Example 1:**

```
Input: "()"
Output: True
```

**Example 2:**

```
Input: "(*)"
Output: True
```

**Example 3:**

```
Input: "(*))"
Output: True
```

**Note:**

1. The string size will be in the range [1, 100].

---

Intution and algorithm:

1. maintain two stack of int open and ast
2. open: stores index where it encountered a '('

   ast : stores index where it encountered a '*'

3. traverse the string s:

- if '(' is enccountered push it onto the open stack
- if '*' is encountered push it onto the ast stack
- if ')' is encountered pop 1 element from open stack, but if open stack is empty pop one element from ast stack, but if this is also empty , return FALSE
- After traversing all the elements , pop elements from open stack , for corresponding elements in ast stack only if index on top of ast > index on top of open stack (to avoid cases like : "*(" )
- if finally open is empty return TRUE, else FALSE

---

CODE:
int n = s.length();

```
    if(n == 0)
        return true;
    if(s[n-1] == '(' || s[0] == ')')
        return false;
 stack<int> open,ast;
  for(int i=0;i<n;i++)
  {

      if(s[i] == ')')
      {
        if(open.empty() && ast.empty())
            return false;
        else if(open.empty()){
            ast.pop();
        }
         else
         {
             open.pop();
         }
      }
      else if(s[i] == '*')
      {
         ast.push(i);
      }
      else{
         open.push(i);
      }
  }
    while(!open.empty() && !ast.empty()){
        if(open.top() > ast.top())
            return false;
        open.pop();
        ast.pop();
    }
    return open.empty();
}
```