Valid parentheses

| Come as the constraint constraint constraints of the constraints and constraints as constraints and constraints. Dempte 10 D | 20. Valid Parentheses |
|--|--|
| Low tracks must be been good of the same one 1. Construction and the construction of the same one 1. Construction and the construction of the same one Longite to Longite to the construction of the same one Longite to the construction of the same of t | Easy 🖒 20546 🖓 1279 ♡ Add to List [ʃ] Share |
| 1. Does broad may be considered to the constitution of the constit | Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. |
| Low constraint to cool in the control real. Low constraint to so the state of the same type. Framps to the specific control real to the same type. Low control real to the same type. Low control real to the same type. Low control real type control real type control real type control real type. Low control real type control real t | An input string is valid if: |
| Stay controved has contended good based of the serve type. Example 1: States: 1 to 1000 Example 2: States: 1 to 1000 Example 3: DP 970 CT - Is your 1 to 1000 Example 3: Stay controlled by the controlle | |
| Tomple 1: Comple 1: Comple 1: Comple 2: Comple 3: Comple 3: Comple 3: Comple 3: Comple 4: Comple 4: Comple 4: Comple 4: Comple 5: Comple 5: Comple 5: Comple 6: | |
| Tample 1: Comple 1: Comple 2: Comple 3: Comple 3: Comple 4: Comple 3: Comple 4: Comple 4: Comple 4: Comple 4: Comple 4: Comple 5: Comple 5: Comple 5: Comple 6: | |
| Example 2: Complete Complete | Example 1: |
| Dample? Description The very Simple we will be using stack which will stone opening Borackets. S= "()[]" The halds opening bracket we will push it to stack to stack then compone it with stack top() if they one cosonect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' Stack top() == '[']' - pop() Else return folse: | |
| Depart to the Complete the control of the control o | |
| Descriptions ADDOCT: It's very simple we will be using stack which will store opening Brackets S= "()[]" Ch=S[0] The holds opening bracket we will push it to stack Ch=S[1] Now Ch holds closing bracket then compone it with stack top() if they one cornect order simply pop stack and move to next element. Ch=S[2] Ch=S[3] if (ch=='3') C C Stack top()=='(1')-pop() Else return folse: | Example 2: |
| This very simple we will be using stack which will store opening Benackels. S="()[]" The body opening bracket we will push it to stack to be considered to be simply pop stack to be stack top!) if they one considered onder simply pop stack to be set to be next element. Ch = S[] Ch = S[] Ch = S[] I wow ch holds closing bracket then campore it with stack top!) if they one considered onder simply pop stack to be set to be next element. Ch = S[] Ch | |
| App 970 CT: It's very simple we will be using stack which will store opening Borackets. S="()[]" Ch=S[0] T) Ch halds opening bracket we will push it to stack (h=S[1]) Now Ch traids closing bracket then compose it with stack. top () if they one coonect order simply pop stack and move to next element. Ch=S[2] Ch=S[3] if (ch="3")) Else return folse: | |
| App 970 Ch: The very simple we will be using stack which will store opening Brackets. S="()[]" Ch=S[0] The holds opening bracket we will push into stack. Ch=S[] Now Ch holds closing bracket then compose it with stack. top() if they one connect conden simply pop stack. One move to next element. Ch=S[2] Ch=S[3] if (ch="3")) Stack top()=c"(")-pop() Plse return folse: | |
| It's very simple we will be using stack which will store opening Brackets. S= ((()[])?) Ch=S[0] The holds opening bracket we will push it to stack Ch=S[] Now Ch holds closing bracket then compone it with stack top() if they one cognect order simply pop stack and move to next element. Ch=S[3] if (ch==']? If stack top()== ([]?-pop()) Plse return folse: | |
| It's very simple we will be using stack which will store opening Brackets. S= ((()[])?) Ch=S[0] The holds opening bracket we will push it to stack (h=S[1]) Now (h holds closing bracket then compone it with stack top() if they one cognect order simply pop stack and move to next element. Ch=S[2] Ch=S[3] if (ch==']) Stack top()== ([])-pop() Else seturn folse: | |
| It's very simple we will be using stack which will store opening Brackets. S= ((()[])?) Ch=S[0] The holds opening bracket we will push it to stack (h=S[1]) Now (h holds closing bracket then compone it with stack top() if they one cognect order simply pop stack and move to next element. Ch=S[2] Ch=S[3] if (ch==']) Stack top()== ([])-pop() Else seturn folse: | |
| It's very simple we will be using stack which will store opening Brackets. S= ((()[])?) Ch=S[0] The holds opening bracket we will push it to stack Ch=S[] Now Ch holds closing bracket then compone it with stack top() if they one cognect order simply pop stack and move to next element. Ch=S[3] if (ch==']? If stack top()== ([]?-pop()) Plse return folse: | |
| It's very simple we will be using stack which will store opening Brackets. S= ((()[])?) Ch=S[0] The holds opening bracket we will push it to stack Ch=S[] Now Ch holds closing bracket then compone it with stack top() if they one cognect order simply pop stack and move to next element. Ch=S[3] if (ch==']? If stack top()== ([]?-pop()) Plse return folse: | Γ Γ Γ Γ Γ Γ Γ |
| S= "(()[]") Ch holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then campore it with Stack. top() if they one cornect order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']") stack top()== '["]"-pop() else return jolse: | |
| S= "(()[]") Ch holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then campore it with Stack. top() if they one cornect order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']") stack top()== '["]"-pop() else return jolse: | |
| S= "(()[]") Ch holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then campore it with Stack. top() if they one cornect order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']") stack top()== '["]"-pop() else return jolse: | This was Cincle we ill to using a took will |
| S= "(()[]") Ch holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then campore it with Stack. top() if they one cornect order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']") stack top()== '["]"-pop() else return jolse: | Is very simple we will be osing stack which will |
| S= "(()[]") Ch holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then compone it with Stack. top() if they one cosnect order simply pop stace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']") stack top()== '["]"-pop() else return jolse: | stone opening Bonackels. |
| Ch=S[0] T) Ch halds opening bracket we will push it is stack Ch=S[1] Now Ch halds closing bracket then compone it with Stack top() if they one correct order simply pop stack and move to next element. Ch=S[2] Ch=S[3] if (ch=='] 1) Stack top()== (z')-pop() Else return jolse: | |
| The holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cognect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack. top() == '['] - pop() Else return folse; | S= "()LJ" |
| The holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cognect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack. top() == '['] - pop() Else return folse; | |
| The holds opening bracket we will push it to stack Ch = S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cognect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack. top() == '['] - pop() Else return folse; | ch = S[o] |
| (n=S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cosorect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack.top()== '[']-pop() Else return folse: | |
| (n=S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cosorect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack.top()== '[']-pop() Else return folse: | T) Ch halds on ening bracket we will |
| (n=S[1] Now Ch holds closing bracket then compose it with stack. top() if they one cosorect order simply pop stack and move to next element. Ch = S[2] Ch = S[3] if (ch == ']' stack.top()== '[']-pop() Else return folse: | Joush it to stack |
| Now Ch halds closing bracket then compose it with stack. top () if they one correct order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']') Stack.top()== '[']-pop() Else return folse: | S-011 45/30 55/0 41 |
| Now Ch halds closing bracket then compose it with stack. top () if they one correct order simply popstace and move to next element. Ch = S[2] Ch = S[3] if (ch == ']') Stack.top()== '[']-pop() Else return folse: | CD = S[1] |
| ond move to next element. Ch = S[2] Ch = S[3] iy (ch == ']'] stack.top()== '[']-pop() Else setusn jolse: | |
| ond move to next element. Ch = S[2] Ch = S[3] iy (ch == ']'] stack.top()== '[']-pop() Else setusn jolse: | No. 16 holds closing be at that they compare it with |
| ond move to next element. Ch = S[2] Ch = S[3] iy (ch == ']'] stack.top()== '[']-pop() Else setusn jolse: | Ctook I was absing walker then compose it with |
| ond move to next element. Ch = S[2] Ch = S[3] iy (ch == ']']) Stack top()== '[']-pop() Else setusn jolse: | Stack top () if they one costrect order simply pop sta |
| Ch = S[2] $Ch = S[3] if (Ch = = '3') if S = (E') - pop()$ $S = S = S[3] otherwise otherw$ | · |
| ch=S[3] ij(ch==']) [stack.top()== '[]-pop() Use return jolse; | and move to next element. |
| ch=S[3] ij(ch==']) [stack.top()== '[]-pop() Use return jolse; | |
| Use return jobse: | Ch = SL2J |
| Use return jobse: | |
| Use return jobse: | $Ch = S [3] \hat{U} (Ch = = ']^2 II$ |
| Use return jobse: | $stack.ton()==(r^2)=000()$ |
| | 54 54 CK 41 4p 37 |
| | fle return labe |
| Afton loop ends if our stack remains empty | Cose or of only Jose , |
| my on loop ends by our significants empty | Alter losso and il and thanks a mains |
| | my or loop ends by our sold in stemoins empty |
| Oreting in Tours also lake | |

```
loop(i-0=s.size()){

(h = S[i]

iy (opening bonacket) push();

else

iy (Check iy in correct onder)

Pop()

else False

S

iy (empty()) Towe

else False.
```

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> stk;
        for(auto ch: s){
            if(ch=='(' || ch=='[' || ch=='{') stk.push(ch);
            else{
                if(!stk.empty()){
                    if( (ch==')' && stk.top()=='(') || (ch==']' && stk.top()=='[') || (ch=='}' && stk.top()=='{')}
                        stk.pop();//Correct Order
                    }
                    else{
                        return false;
                else return false;
        if(stk.empty()) return true;
        else return false;
};
```

1. C - O (n) n: length g 5. C - O (n) S