Balanced Parenthesis.

I/p : ([ )]

O/p :- Not valid

I/p : { } ([( )])

O/p : Valid.

→ if in ques this is valid too, then just do x++ & x--; at last x==0

x++ → (, [, {
x-- →),], }

Code:-

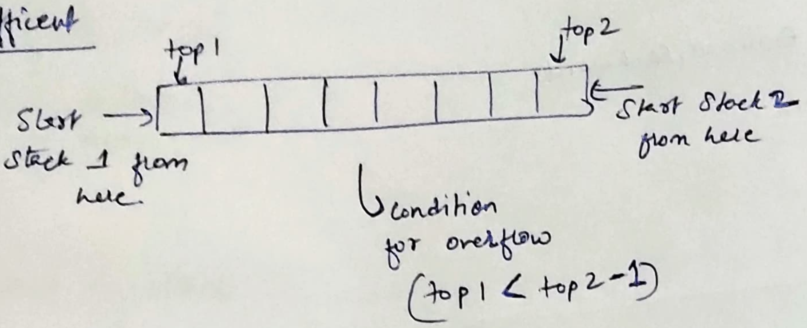
```
boolean matching(char a, char b)
{ return((a=='(' && b==')')||
         (a=='[' && b==']')||
         (a=='{' && b=='}'));
}

boolean isBalanced(String str)
{ Deque<character> s = new ArrayDeque<>();
  for(int i=0; i<str.length(); i++)
  { char x = str.charAt(i);
    if(x=='(' || x=='{' || x=="[")
      { s.push(x); }
    else { if(s.isEmpty()== false) { return false; }
      else if(matching(s.peek(),x)== false){ return false;}
      else { s.pop();}
    }
  } return (s.isEmpty()==true);
}
```

2 or more Stacks in an array.

Naive → divide the array into 2 parts from middle.

Efficient



Stock Span Problem.

→ Span on a day → (Including that day = 1) + (No: of days with less value than it on left side that are consecutive)

→ Eg:- [30, 20, 25, 28, 27, 29]

1 1 2 3 1 2

① → Naive Solution → $O(n^2)$

→ Traversing left side of every array element.

② Code

```
→ void printSpan (int arr[], int n)
  { for (int i=0; i<n; i++)
     { int span=1
        for (int j=i-1 ; j>=0 && arr[j] <= arr[i]; j--)
          { span++; }
        print(span + " ");
     }
  }
```

② Efficent solution

```
void printSpan(int arr[], int n)
{ Stack S;
  s.push(0);
  print(1);

  for(int i=1; i<n; i++)
  { while(s.isEmpty()==false; && arr[s.top()]≤ arr[i])   ] Removing all smaller items.
    {
       s.pop();
    }
    span = s.isEmpty() ? i+1 : i-s.top();
    print(span);
    s.push(i);
  }
}
```

Span of item = (Index of curr item) − (Index of prev greater elem)

Perious Greater Element. (variation of Stock span).

I/p → [15, 10, 18, 12, 4, 6, 2, 8]
-1 15 -1 18 12 12 6 12
O/p →

I/p → {8, 10, 12}
O/p → -1 -1 -1

P.T.O →

Naive Sol → Use two loops , traverse from right ~~for~~ each element.
O(n2)

Efficent Sol → print prevGreater (int arr[], int n)

```
{  Stack <int> s;

   s.push(arr[0]);

  for(int i=0; i<n; i++)
  {
     while(s.isempty == false && s.top() ≤ arr[i])
     { s.pop();
     }
     int pg = (s.empty()) ? -1 : s.top();
     cout << pg << "  ";
     s.push (arr[i]);
  }
}
```