```
1 8
5 12
14 19
22 28
25 27
27 30
```

1-12

14-19

22-30

$$\left.\begin{array}{ll} 1 & 8 \\ 5 & 12 \end{array}\right] \quad 1-12$$

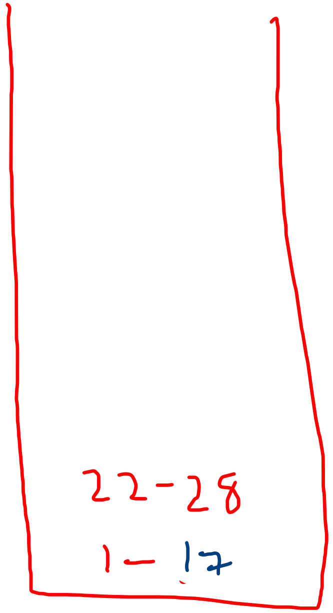
$$\left.\begin{array}{ll} 14 & 19 \end{array}\right] \quad 14-19$$

$$\left.\begin{array}{ll} 25 & 27 \\ 27 & 30 \\ 22 & 28 \end{array}\right] \quad 22-30$$

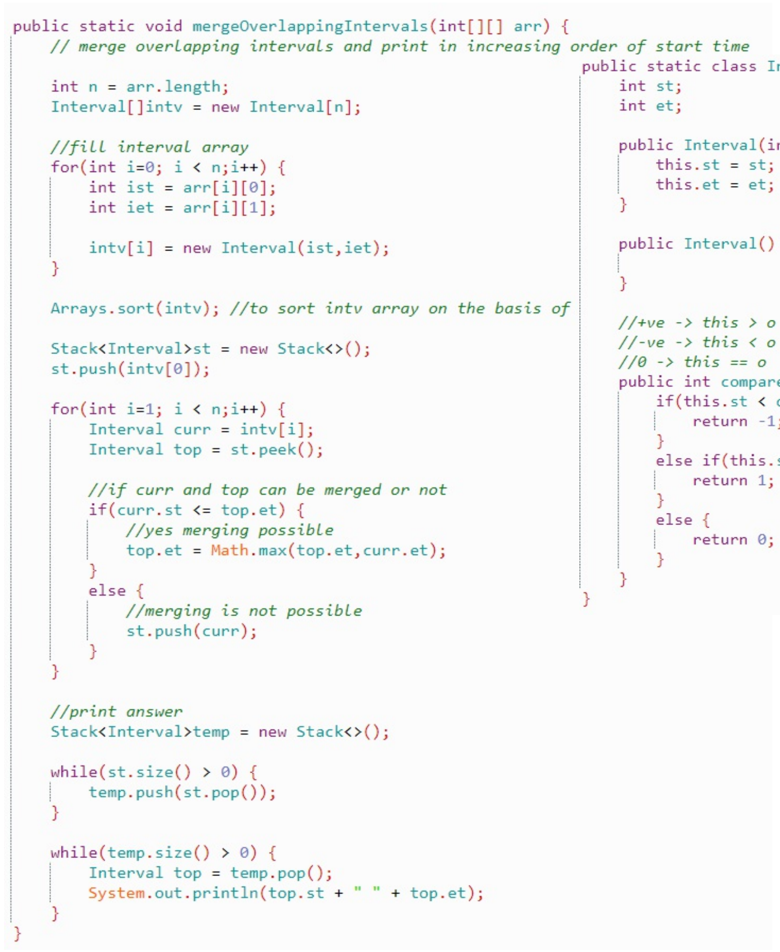
5	12
14	19
25	27
27	30
22	28
1	8
11	17

(i)

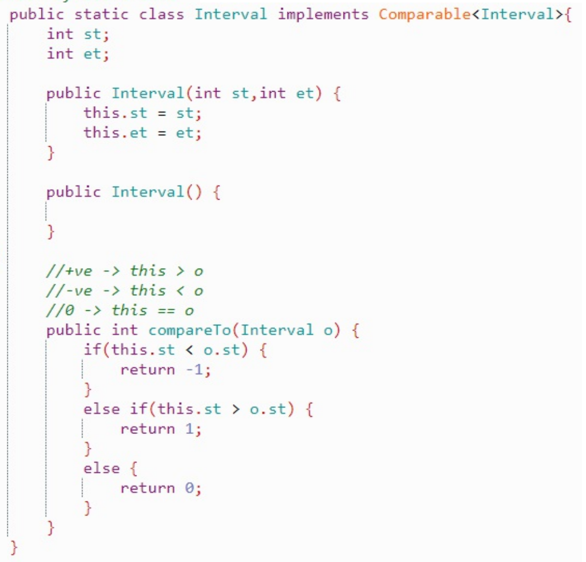
	1	8
	5	12
	11	17
	14	16
$\longrightarrow$	22	28
	25	27
	27	30



C = 22 –28



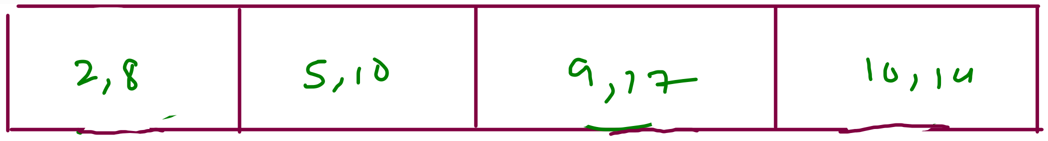
```
public static void mergeOverlappingIntervals(int[][] arr) {
    // merge overlapping intervals and print in increasing order of start time

    int n = arr.length;
    Interval[]intv = new Interval[n];

    //fill interval array
    for(int i=0; i < n;i++) {
        int ist = arr[i][0];
        int iet = arr[i][1];

        intv[i] = new Interval(ist,iet);
    }

    Arrays.sort(intv); //to sort intv array on the basis of

    Stack<Interval>st = new Stack<>();
    st.push(intv[0]);

    for(int i=1; i < n;i++) {
        Interval curr = intv[i];
        Interval top = st.peek();

        //if curr and top can be merged or not
        if(curr.st <= top.et) {
            //yes merging possible
            top.et = Math.max(top.et,curr.et);
        }
        else {
            //merging is not possible
            st.push(curr);
        }
    }

    //print answer
    Stack<Interval>temp = new Stack<>();

    while(st.size() > 0) {
        temp.push(st.pop());
    }

    while(temp.size() > 0) {
        Interval top = temp.pop();
        System.out.println(top.st + " " + top.et);
    }
}
```

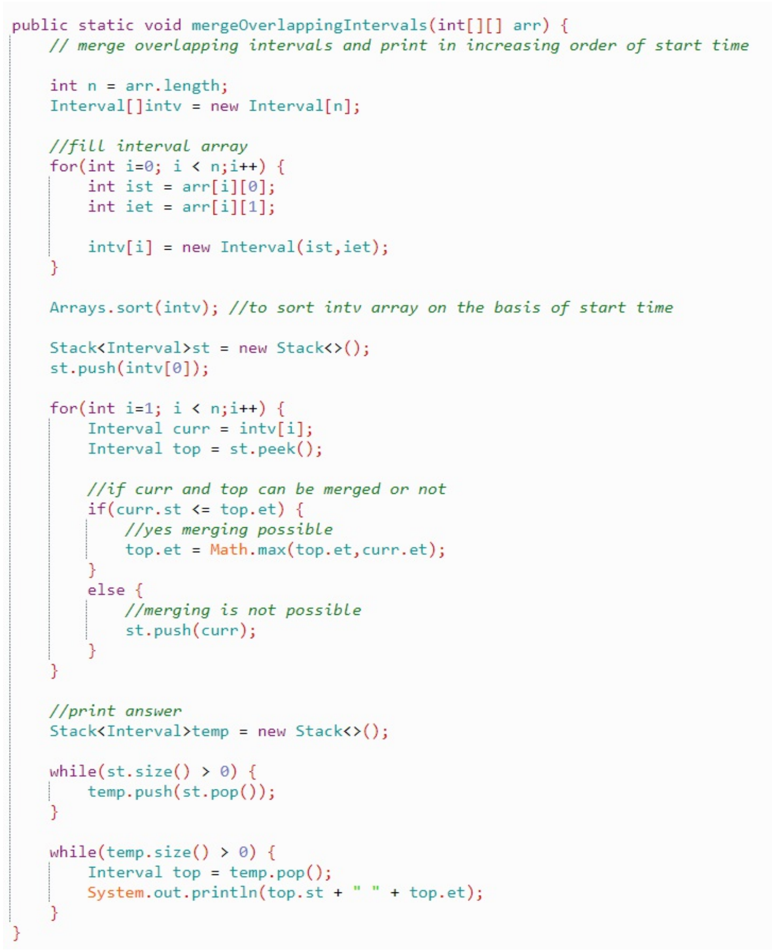


```
public static class Interval implements Comparable<Interval>{
    int st;
    int et;

    public Interval(int st,int et) {
        this.st = st;
        this.et = et;
    }

    public Interval() {

    }

    //+ve -> this > o
    //-ve -> this < o
    //0 -> this == o
    public int compareTo(Interval o) {
        if(this.st < o.st) {
            return -1;
        }
        else if(this.st > o.st) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```

this = 5,12

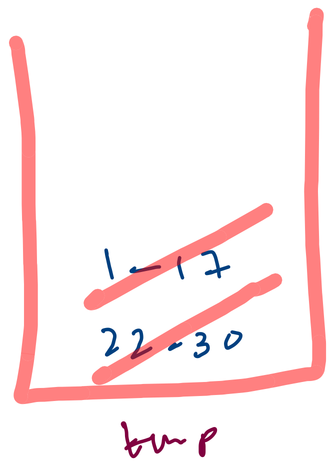
o = 10,1

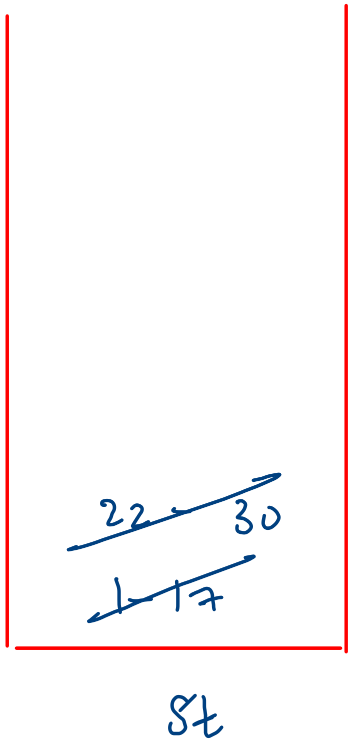




```java
public static void mergeOverlappingIntervals(int[][] arr) {
    // merge overlapping intervals and print in increasing order of start time

    int n = arr.length;
    Interval[]intv = new Interval[n];

    //fill interval array
    for(int i=0; i < n;i++) {
        int ist = arr[i][0];
        int iet = arr[i][1];

        intv[i] = new Interval(ist,iet);
    }

    Arrays.sort(intv); //to sort intv array on the basis of start time

    Stack<Interval>st = new Stack<>();
    st.push(intv[0]);

    for(int i=1; i < n;i++) {
        Interval curr = intv[i];
        Interval top = st.peek();

        //if curr and top can be merged or not
        if(curr.st <= top.et) {
            //yes merging possible
            top.et = Math.max(top.et,curr.et);
        }
        else {
            //merging is not possible
            st.push(curr);
        }
    }

    //print answer
    Stack<Interval>temp = new Stack<>();

    while(st.size() > 0) {
        temp.push(st.pop());
    }

    while(temp.size() > 0) {
        Interval top = temp.pop();
        System.out.println(top.st + " " + top.et);
    }
}
```

0 ->	1	8
1 ->	5	12
2 ->	11	17
3 ->	14	16
4 ->	22	28
5 ->	25	27
6 ->	27	30





1-17

22-30

e.g.
d -> 21
i -> 12
ddd -> 4321
iii -> 1234
dddiddd -> 43218765
iiddd -> 126543

1 2 3 4
i i i

d -> dec
i -> inc

4 3 2 1
d d d

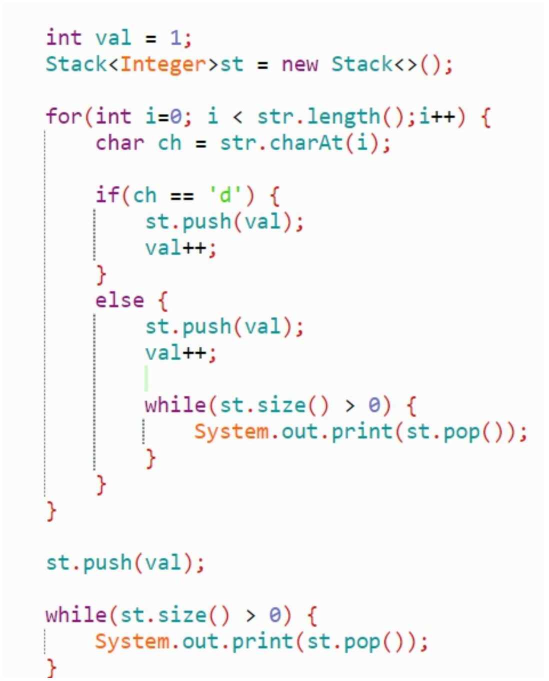
pattern ≤ 8
ans -> pat. len + 1
└ 1 to 9
Smallest

3 2 1 7 6 5 4 8
d d i d d d i

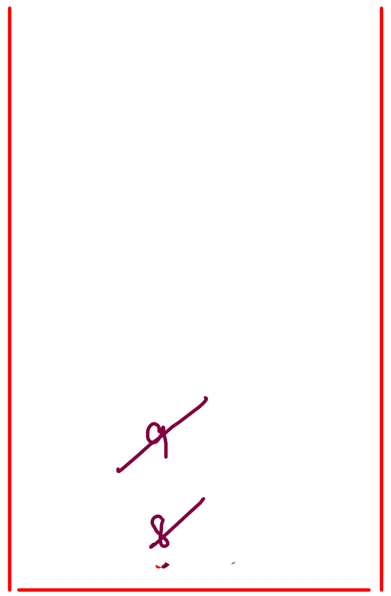
3 2 1 7 6 5 4 8

4 3 2 1 6 5 7 9 8

d d d i d i i d



```
int val = 1;
Stack<Integer>st = new Stack<>();

for(int i=0; i < str.length();i++) {
    char ch = str.charAt(i);

    if(ch == 'd') {
        st.push(val);
        val++;
    }
    else {
        st.push(val);
        val++;

        while(st.size() > 0) {
            System.out.print(st.pop());
        }
    }
}

st.push(val);

while(st.size() > 0) {
    System.out.print(st.pop());
}
```

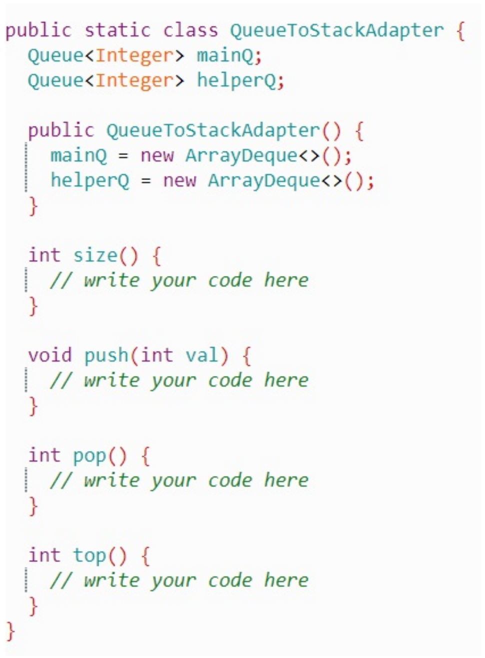


val = 9

```
if( ch = = 'd') {
      st.push(val);
      val++;
}
else {
      st.push(val);
      val++;
      print(st);
}
```

Queue to Stack adapter

push efficient.



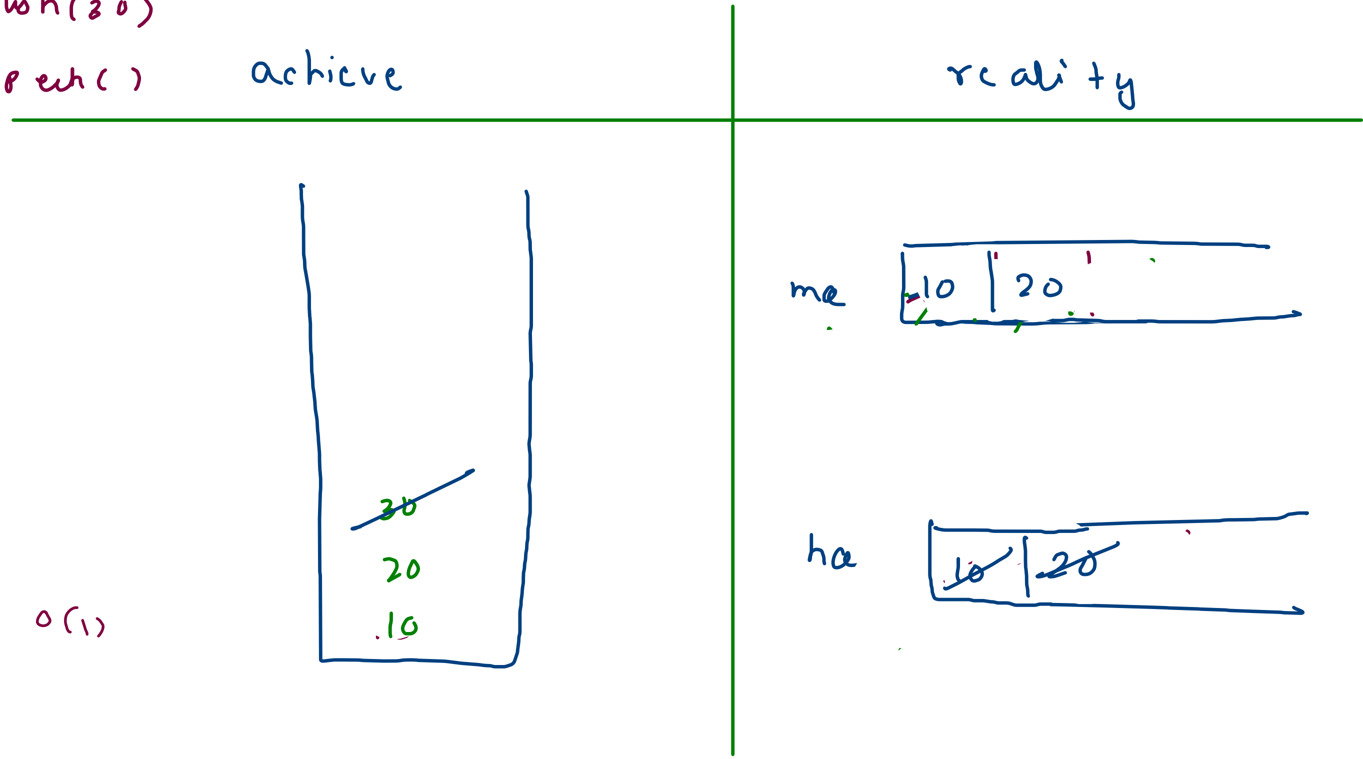
```java
public static class QueueToStackAdapter {
  Queue<Integer> mainQ;
  Queue<Integer> helperQ;

  public QueueToStackAdapter() {
    mainQ = new ArrayDeque<>();
    helperQ = new ArrayDeque<>();
  }

  int size() {
    // write your code here
  }

  void push(int val) {
    // write your code here
  }

  int pop() {
    // write your code here
  }

  int top() {
    // write your code here
  }
}
```

St.push(10)

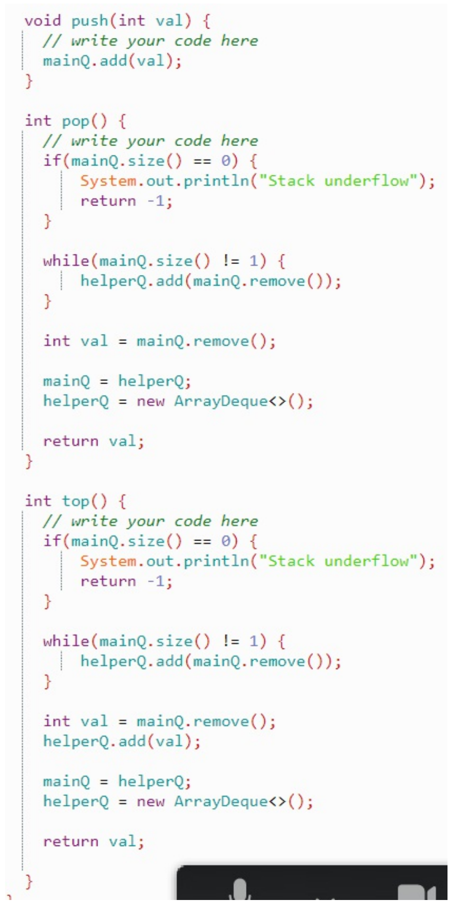
St.push(20)

St.push(30)

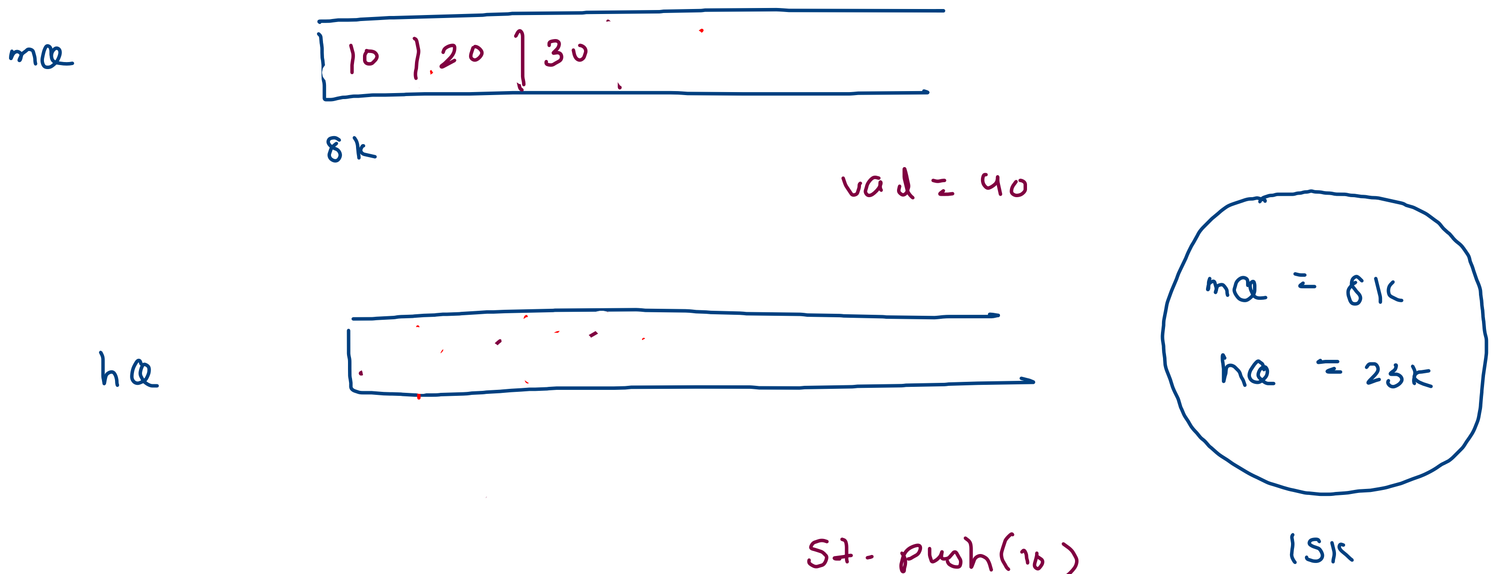
St.peek( )



Stack utility — pop
— push O(1)
— top



```java
void push(int val) {
    // write your code here
    mainQ.add(val);
}

int pop() {
    // write your code here
    if(mainQ.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }

    while(mainQ.size() != 1) {
        helperQ.add(mainQ.remove());
    }

    int val = mainQ.remove();

    mainQ = helperQ;
    helperQ = new ArrayDeque<>();

    return val;
}

int top() {
    // write your code here
    if(mainQ.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }

    while(mainQ.size() != 1) {
        helperQ.add(mainQ.remove());
    }

    int val = mainQ.remove();
    helperQ.add(val);

    mainQ = helperQ;
    helperQ = new ArrayDeque<>();

    return val;
}
```



St. push(10)

St. push(20)

St. push(30)

St. push(40)

St. peek()

St. pop()

Queue to stack adapter

pop - efficient.

$$\begin{tikzcd} & & pop \\ Stack & utility \arrow[ur, dash] \arrow[r, dash] \arrow[dr, dash] & push \\ & & top \end{tikzcd}$$

o(1)

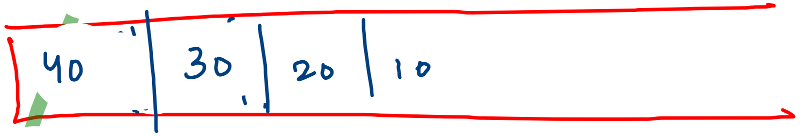
st.push(10)

st.push(20)

st.pu(30)

st.pu(40)

mq



hq

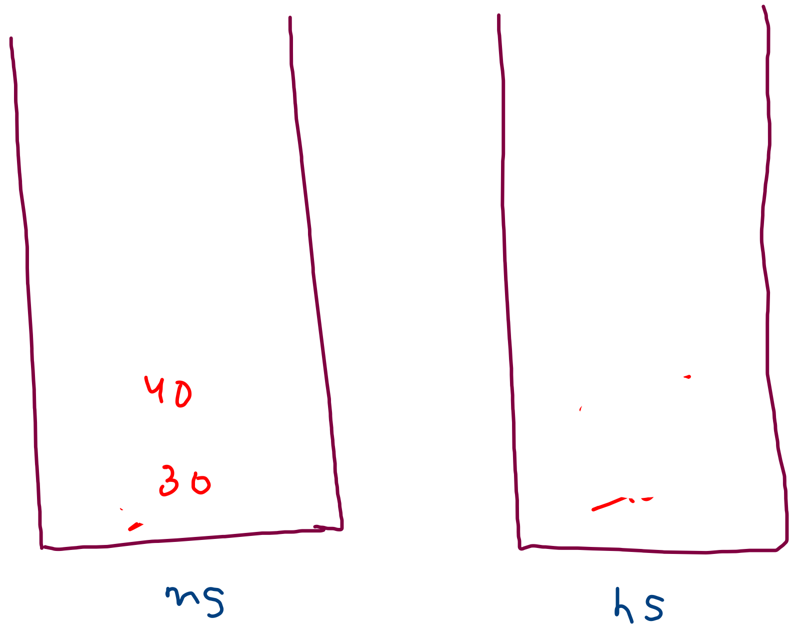
Stack to queue adapter

add efficient

queue utility
- add O(1)
- remove
- peek

20

q.add(10)
q.add(20)
q.add(30)
q.peek()
q.add(40)
q.remove()



Stack to queue adapter

remove efficient

q.add(10)

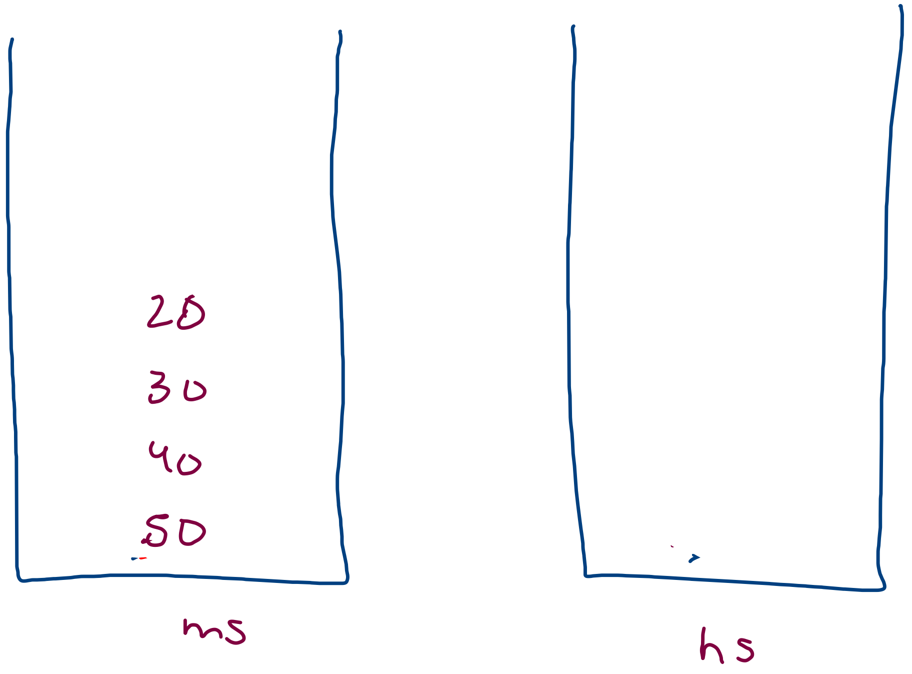
q.add(20)

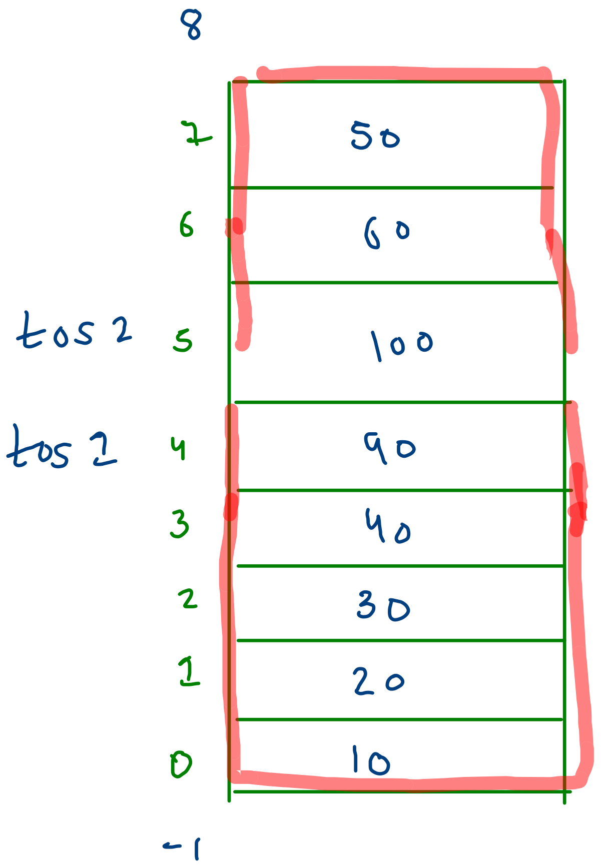
q.add(30)

q.add(40)

q.remove( ) $\rightarrow$ ms.pop( )

q.add(50)





cap = 8

push 1 (10)

push 1 (20)

push 2 (50)

push 1 (30)

push 2 (60)

push 1 (90)

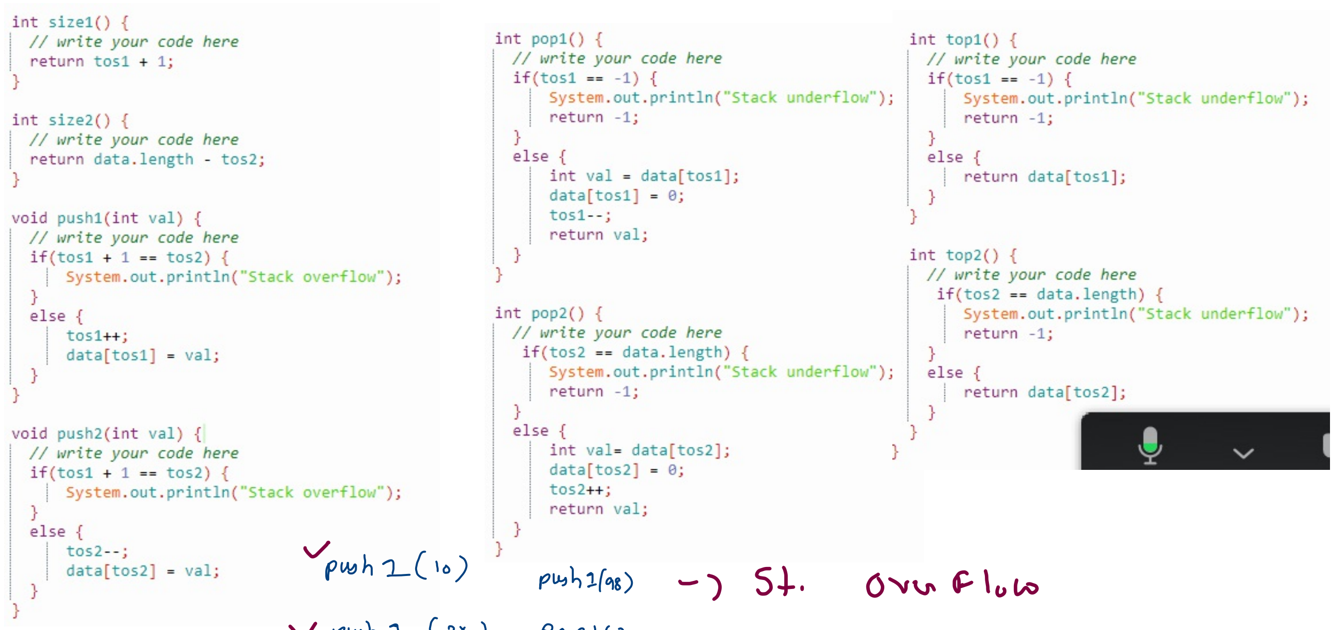
push 1 (90)

push 2 (87)

push 1 (98)

pop 1 ()

pop 2 ()



```java
int size1() {
  // write your code here
  return tos1 + 1;
}

int size2() {
  // write your code here
  return data.length - tos2;
}

void push1(int val) {
  // write your code here
  if(tos1 + 1 == tos2) {
      System.out.println("Stack overflow");
  }
  else {
      tos1++;
      data[tos1] = val;
  }
}

void push2(int val) {
  // write your code here
  if(tos1 + 1 == tos2) {
      System.out.println("Stack overflow");
  }
  else {
      tos2--;
      data[tos2] = val;
  }
}
```

```java
int pop1() {
  // write your code here
  if(tos1 == -1) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      int val = data[tos1];
      data[tos1] = 0;
      tos1--;
      return val;
  }
}

int pop2() {
  // write your code here
   if(tos2 == data.length) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      int val= data[tos2];
      data[tos2] = 0;
      tos2++;
      return val;
  }
}
```

```java
int top1() {
  // write your code here
  if(tos1 == -1) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      return data[tos1];
  }
}

int top2() {
  // write your code here
   if(tos2 == data.length) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      return data[tos2];
  }
}
}
```

✓ push 1 (10)   push1(98) → St. overflow

✓ push 1 (20)   pop1() →

✓ push 2 (50)   pop2( ) →

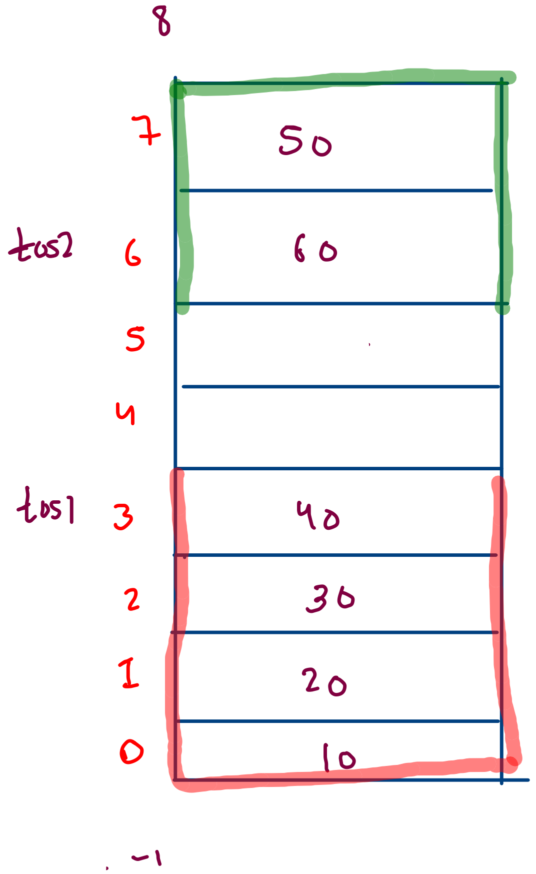
✓ push 1 (30)

✓ push 2 (60)

✓ push 1 (40)

✓ push 2 (90)

✓ push2(87)



8
7
50
tos2
6
60
5
4
tos1
3
40
2
30
1
20
0
10
-1