Merge two sorted arrays 7

$n = 4$

$A = [1, 3, 3, 7]$

$m = 4$

$B = [2, 4, 4, 8]$

i

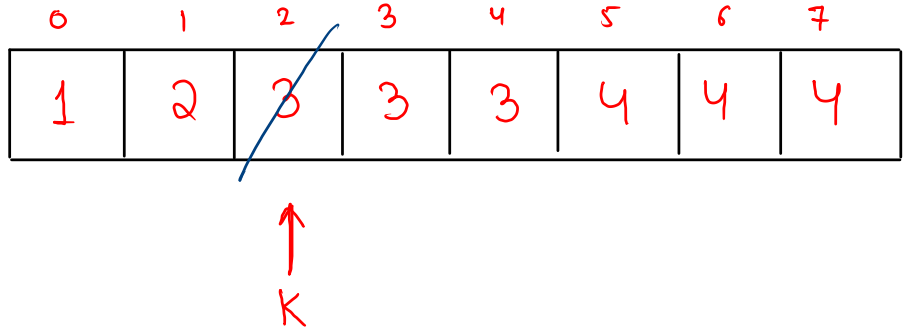
j

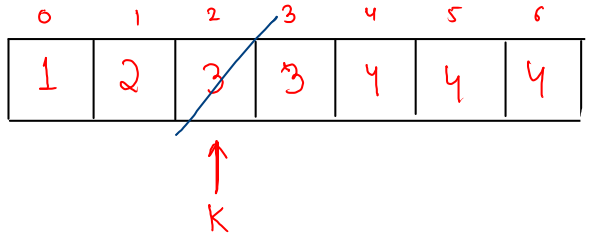
loop until $i < n$ && $j < m$

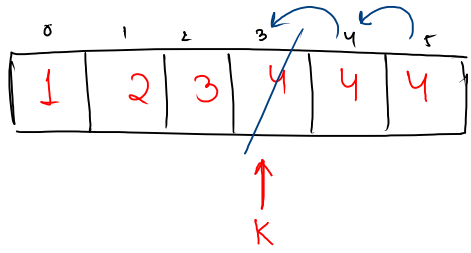
AL = | 1 | 2 | 3 | 3 | 4 | 4 | 7 | 8 |

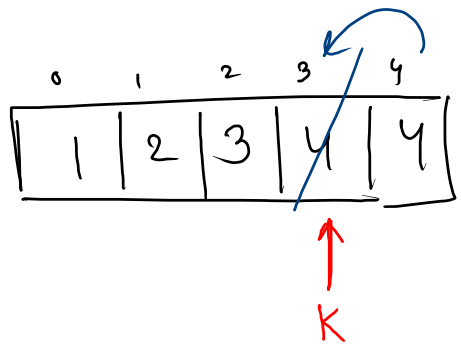
AL = | 1 | 2 | 3 | 4 | 7 | 8 |

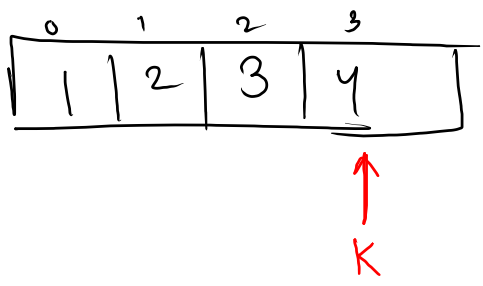
AL =


0 1 2 3 4 5 6 7
1 2 3 3 3 4 4 4
K


0 1 2 3 4 5 6
1 2 3 3 4 4 4
K


0 1 2 3 4 5
1 2 3 4 4 4
K


0 1 2 3 4
1 2 3 4 4
K


0 1 2 3
1 2 3 4
K

```
n = arr.size();

k = 0
while( k < n-1 ) {
    if( arr.get(k) == arr.get(k+1) ){
        arr.remove(k);
    } else {
        k++
    }
}
```

1)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] A = new int[n];
    for (int i = 0; i < n; i++) {
        A[i] = scn.nextInt();
    }
    int m = scn.nextInt();
    int[] B = new int[m];
    for (int i = 0; i < m; i++) {
        B[i] = scn.nextInt();
    }
    merge2Arrays(A, n, B, m);
}
```

2)

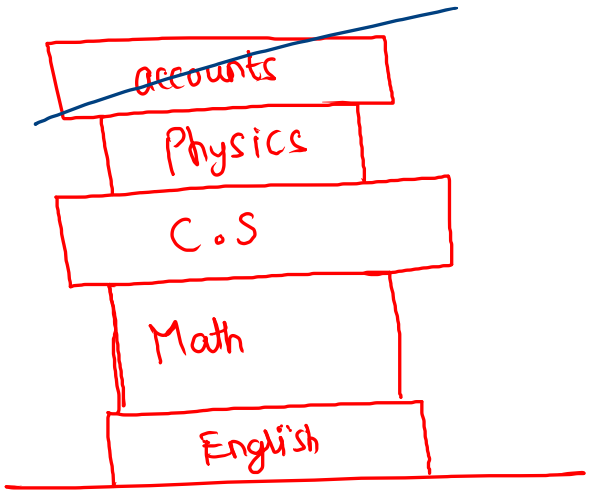
```java
public static void merge2Arrays(int[] A, int n, int[] B, int m) {
    ArrayList<Integer> ans = new ArrayList<>();
    int i = 0;
    int j = 0;
    while ( i < n && j < m ) {
        if ( A[i] < B[j] ) {
            ans.add( A[i] );
            i++;
        } else {
            ans.add( B[j] );
            j++;
        }
    }
    while ( i < n ) {
        ans.add( A[i] );
        i++;
    }
    while ( j < m ) {
        ans.add( B[j] );
        j++;
    }
    // now how to handle duplicacy
    int k = 0;
    while ( k < ans.size() - 1 ) {
        if ( ans.get(k) == ans.get(k + 1) ) {
            ans.remove(k);
        } else {
            k++;
        }
    }
    // print
    for (Integer a : ans) {
        System.out.print(a + " ");
    }
    System.out.println();
}
```

T.C = O(m+n)

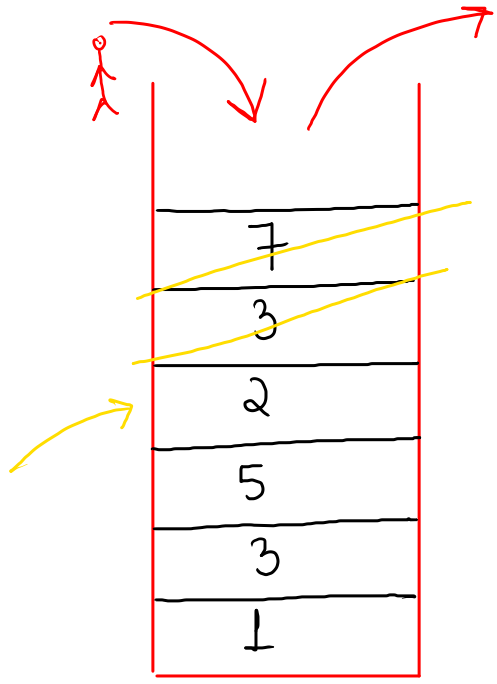
or O(ans.size())

S.C = O(m+n)

⇒ Stack [LIFO :- last in first out]



stack



Note :- stack doesnot have indexing

Syntax `Stack<Integer> st = new Stack<>();`

triangular brackets

⇒ Inbuilt functions

- To add an element in stack :- `st.push(value)`
- To remove an element from stack :- `st.pop()`
- To get the top element of stack :- `st.peek()`
- To get the size of stack :- `st.size()`
- To check if stack is empty :- `st.isEmpty()`

Stack Syntax Learning

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    Stack<Integer> st = new Stack<>();
    int t = scn.nextInt();
    for (int i = 0; i < t; i++) {
        int c = scn.nextInt();
        if ( c == 1 ) {
            printSize(st);
        } else if ( c == 2 ) {
            removeElement(st);
        } else if ( c == 3 ) {
            int x = scn.nextInt();
            addElement(st, x);
        } else if ( c == 4 ) {
            printTopElement(st);
        }
    }
}
public static void printSize(Stack<Integer> st) {
    System.out.println( st.size() );
}
public static void removeElement(Stack<Integer> st) {
    if ( st.isEmpty() ) {
        System.out.println("-1");
        return;
    }
    int ans = st.pop();
}
public static void addElement(Stack<Integer> st, int x) {
    st.push(x);
}
public static void printTopElement(Stack<Integer> st) {
    if ( st.isEmpty() ) {
        System.out.println("-1");
        return;
    }
    System.out.println( st.peek() );
}
```

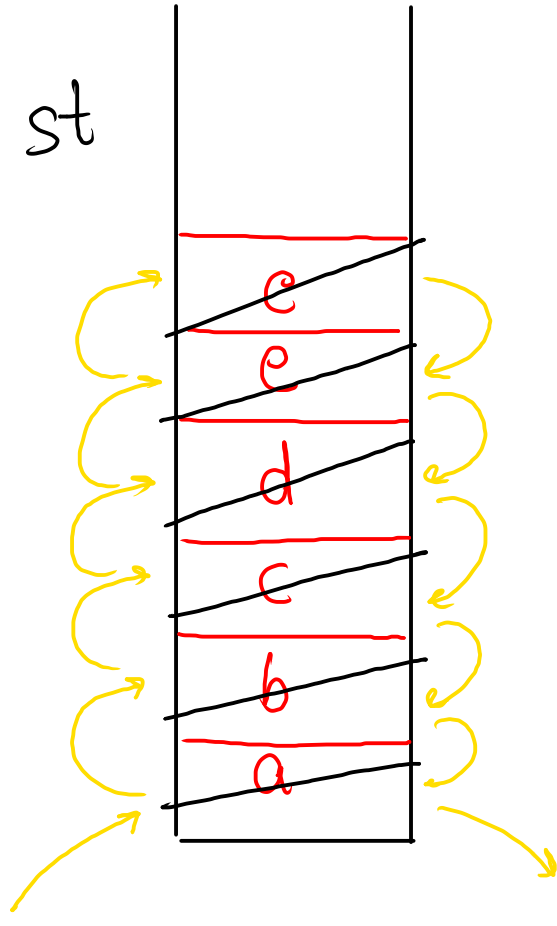
Reverse string

Recursion

str = " a b c d e e "

0 1 2 3 4 5

ans = " e e d c b a "



Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    System.out.println(reverseString(str));
}
public static String reverseString(String str) {
    Stack<Character> st = new Stack<>();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        st.push( ch );
    }

    String ans = "";
    while ( st.size() > 0 ) {
        char ch = st.peek();
        ans = ans + ch;
        st.pop();
    }
    return ans;
}
```

T.C = O(N)

S.C = O(N)