String Builder

→ String in Java are immutable.

→ str = "bcde"

→ str + = "a"

↳ O(n)

⇒ To overcome above issues → String Builder

↳ sb = "abcdef"
0 1 2 3 4 5

sb = "abckef"

↳ sb = "abcdef"

sb = "abcdefg" → O(1) not O(n)

Syntax of StringBuilder :-

StringBuilder name_of_sb = new StringBuilder();

Ex →

StringBuilder sb = new StringBuilder(); → ""

StringBuilder sb = new StringBuilder("abc"); → abc

```
StringBuilder sb = new StringBuilder("abc");
System.out.println(sb);
```

(1) Printing → sysout(sb);

(a) using loops: — for (int i = 0; i < sb.length(); i++)

{

sysout (sb.charAt(i));

}

(2) length of sb.

↳ sb.length();

(3) Update index value in sb.

setCharAt(index, newvalue);

sb = "abc"
0 1 2

sb.setCharAt(1, 'd');

↳ sb = "adc";

(4) append → add the group of characters to sb.

sb.append("efgh"); → sb = "adcefgh"
0 1 2 3 4 5 6

(5) Delete a character from sb

sb.deleteCharAt(validIndex);

sb.deleteCharAt(4)

↳ sb = "adc2gh"
0 1 2 3 4 5

(6) Delete group of characters. 2, 4

sb.delete(startingIndex, endingIndex); → ending index is excluded.

sb.delete(2,4)

↳ 2, 3 →

(7) 0 1 2 3
adgh.

↳ bcdefg → a|bcdefg|h

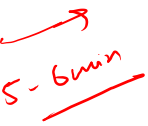
sb.replace(startingIndex, endingIndex, newStrValue);

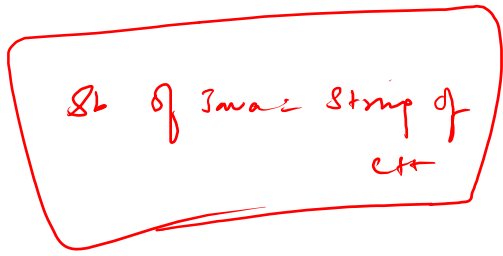
(8) Convert sb to string

toString

Ex → sb.toString();

```java
public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your cl
    StringBuilder sb = new StringBuilder("abc");
    System.out.println(sb);
    System.out.println("Length of String Builder is: " + sb.length());


    for(int i=0;i<sb.length();i++){
        System.out.println(sb.charAt(i));
    }

    sb.setCharAt(1,'d');
    System.out.println("Updated Sb: " + sb);

    sb.append("efgh");
    System.out.println(sb);

    sb.deleteCharAt(4);
    System.out.println(sb);

    sb.delete(2,4);
    System.out.println(sb);

    sb.replace(1,3,"bcdefg");
    System.out.println(sb);

    System.out.println(sb.getClass());
    String str = sb.toString();
    System.out.println(str);
    System.out.println(str.getClass());

}
```





```java
public class Solution {
    public static String toggleCase2(String str){
        StringBuilder sb = new StringBuilder();
        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            if(ch >='a' && ch<='z'){
                sb.append((char)(ch-'a'+'A') + "");
            }else{
                sb.append((char)(ch-'A'+'a') + "");
            }
        }
        return sb.toString();
    }

    public static StringBuilder toggleCase(String str){
        StringBuilder sb = new StringBuilder();
        for(int i=0;i<str.length();i++){
            char ch = str.charAt(i);
            if(ch >='a' && ch<='z'){
                sb.append((char)(ch-'a'+'A') + "");
            }else{
                sb.append((char)(ch-'A'+'a') + "");
            }
        }
        return sb;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String str = scn.next();
        System.out.println(toggleCase2(str));

    }
}
```

5 + 5

↳ Frequent Characters

Priority Queue -

Given a string and an integer k, print the first k high frequency characters. Note: If the frequency of two characters is same then, print characters in lexicographical order.

Input Format

A String An integer number k

Constraints

1<=str.length()<=10000 contains only small case alphabet 0<=k<=str.length()

Output Format

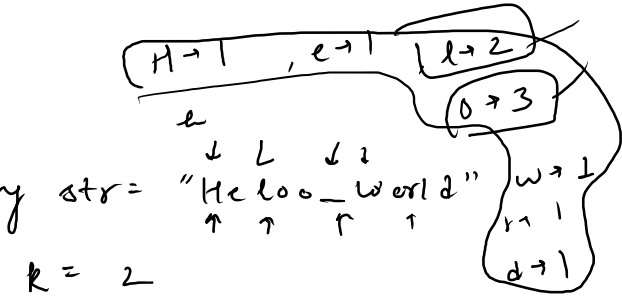
Series of k character in sorted order in single line

Sample Input 0

```
Heloo World
2
```

Sample Output 0

```
o l
```



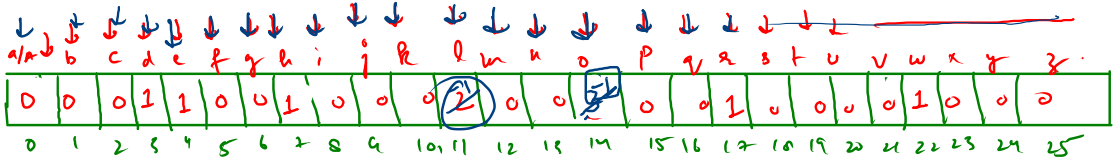
```java
public static void kFrequentCharacters(String str,int k){
    int freqArr [] =  new int[26];
    for(int i=0;i<str.length();i++){
        char ch = str.charAt(i);
        if(ch>='a' && ch<='z'){
            freqArr[ch-'a']++;
        }else if(ch>='A' && ch <='Z'){
            freqArr[ch-'A']++;
        }
    }


    for(int i=1;i<=k;i++){
        int maxFreq = 0;
        int maxFreqIndex = 0;
        char maxCh = 'a';

        for(int j=0;j<26;j++){
            if(freqArr[j] > maxFreq){
                maxFreqIndex = j;
                maxFreq = freqArr[j];
                maxCh = (char)(j+'a');
            }
        }

        System.out.print(maxCh + " ");
        freqArr[maxFreqIndex] = -1;


    }
}
```

Submit

str "Hello - world"
0 1 2 3 4 5 6 7 8 9 10

i = 0 < 11 →

'H' - 'A'



i = 1 < = 2

maxFreq = 0 1 2 3

maxFreqIndex = 0 3 14

maxCh = 'a' 'd' 'o'

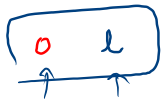
i = 3 < = 2 (F)

i = 2 < = 2

maxFreq = 0 2

maxFreqIndex = 0 11

maxCh = 'a' 'l'



Spiral Display (16 july)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers, representing elements of 2d array a.
4. You are required to traverse and print the contents of the 2d array in form of a spiral.

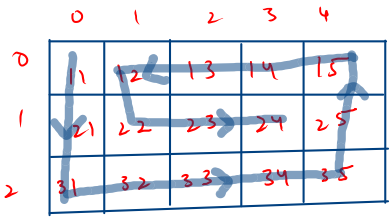
n = 3
m = 5

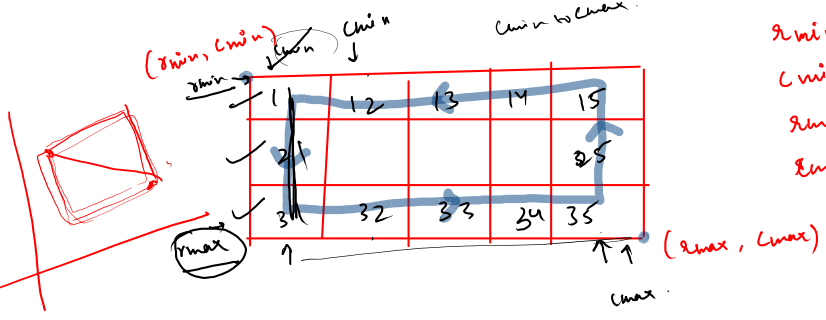
Sample Input 0

```
3
5
11
12
13
14
15
21
22
23
24
25
31
32
33
34
35
```

Sample Output 0

```
11
21
31
32
33
34
35
25
15
14
13
12
22
23
24
```





rmin = 0
cmin = 0
rmax = n-1 | 2
rmax = m-1 | 4

1) downward dirn

```
for (int row = rmin ; row <= rmax; row++)
{
    sysout(arr[row][cmin]);
}
cmin++;
```

2) horizontal (left to right);

```
for (int col = cmin; col <= cmax; col++)
{
    sysout(arr[rmax][col]);
}
```

Spiral Display (16 july)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers, representing elements of 2d array a.
4. You are required to traverse and print the contents of the 2d array in form of a spiral.

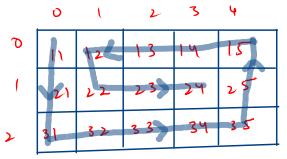
n = 3
m = 5

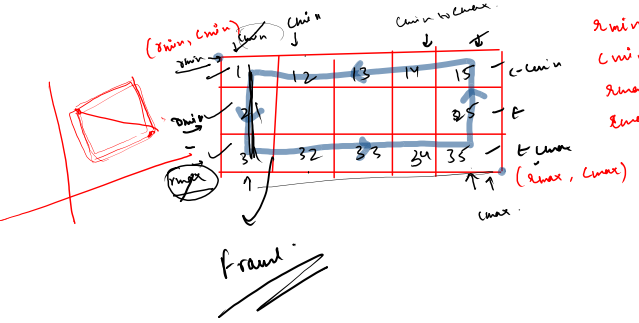
Sample Input 0

```
3
5
11
12
13
14
15
21
22
23
24
25
31
32
33
34
35
```

Sample Output 0

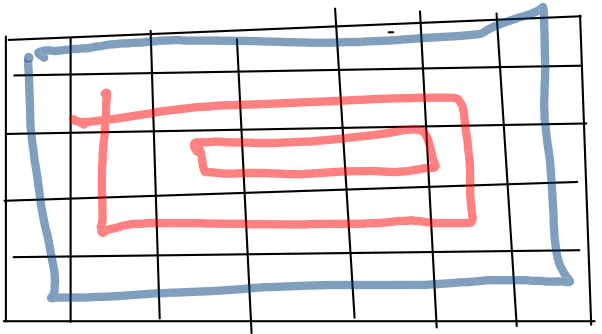
```
11
21
31
32
33
34
35
25
15
14
13
12
```





rmin = 0
cmin = 0
rmax = n-1 | 2
cmax = m-1 | 4

fraud



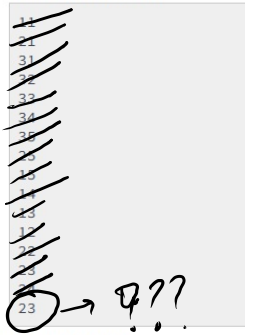
```
while (count < mxn)
{
1) downward dirn
for (int row = rmin ; row <= rmax; row++)
{
    sysr(arr[row][cmin]);
}
cmin++;
2) horizontal ( left to right);
for (int col = cmin; col <= cmax; col++)
{
    sysr (arr[rmax][col]);
}
rmax--;
```

count = 0

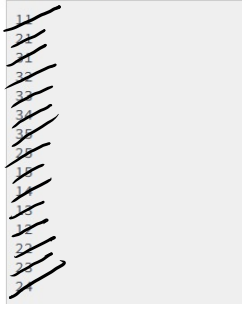
```
3) upward dirn
for (int row = rmax ; row >= rmin;
                        row--)
{
    sysr (arr[row][cmax]);
}
cmax--
4) horizontal ( right to left)
for (int col = cmax; col >= cmin;
                        col--)
{
    sysr (arr[rmin][col]);
}
```

```java
public static void spiralDisplay(int arr[][],int n,int m){
    int rmin = 0;
    int cmin = 0;
    int rmax= n-1;
    int cmax = m-1;

    int count = 0;

    while(count < m*n){
        // downward dirn
        for(int row = rmin;row<=ramx;row++){
            System.out.println(arr[row][cmin]);
            count++;
        }

        cmin++;

        // horizontal dirn (l to r)
        for(int col = cmin;col<=cmax;col++){
            System.out.println(arr[rmax][col]);
            count++;
        }

        rmax--;

        // upward dirn
        for(int row =rmax;row>=rmin;row--){
            System.out.println(arr[row][cmax]);
            count++;
        }

        cmax--;

        for(int col = cmax;col>=cmin;col--){
            System.out.println(arr[rmin][col]);
            count++;
        }

        rmin++;
    }
}
```

Your Output (stdout)

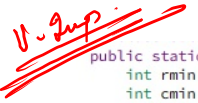


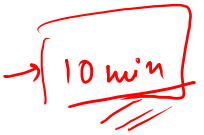
Expected Output

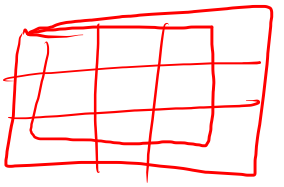


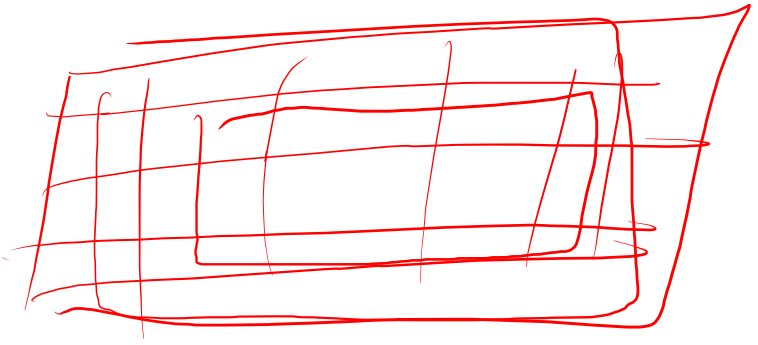
```java
public static void spiralDisplay(int arr[][],int n,int m){
    int rmin = 0;
    int cmin = 0;
    int rmax= n-1;
    int cmax = m-1;

    int count = 0;

    while(count < m*n){
        // downward dirn
        for(int row = rmin;row<=ramx;row++){
            System.out.println(arr[row][cmin]);
            count++;
        }

        cmin++;

        // horizontal dirn (l to r)
        for(int col = cmin;col<=cmax;col++){
            System.out.println(arr[rmax][col]);
            count++;
        }

        rmax--;

        // upward dirn
        for(int row =rmax;row>=rmin;row--){
            System.out.println(arr[row][cmax]);
            count++;
        }

        cmax--;

        for(int col = cmax;col>=cmin;col--){
            System.out.println(arr[rmin][col]);
            count++;
        }

        rmin++;
    }
}
```

```java
public static void spiralDisplay(int arr[][],int n,int m){
    int rmin = 0;
    int cmin = 0;
    int rmax= n-1;
    int cmax = m-1;

    int count = 0;

    while(count < m*n){
        // downward dirn
        for(int row = rmin;row<=rmax && count<m*n;row++){
            System.out.println(arr[row][cmin]);
            count++;
        }

        cmin++;

        // horizontal dirn (l to r)
        for(int col = cmin;col<=cmax && count<m*n;col++){
            System.out.println(arr[rmax][col]);
            count++;
        }

        rmax--;

        // upward dirn
        for(int row =rmax;row>=rmin && count<m*n;row--){
            System.out.println(arr[row][cmax]);
            count++;
        }

        cmax--;

        for(int col = cmax;col>=cmin && count<m*n;col--){
            System.out.println(arr[rmin][col]);
            count++;
        }

        rmin++;
    }
}
```









Toggle and 2 jumps left 1

Problem | Submissions | Leaderboard | Discussions

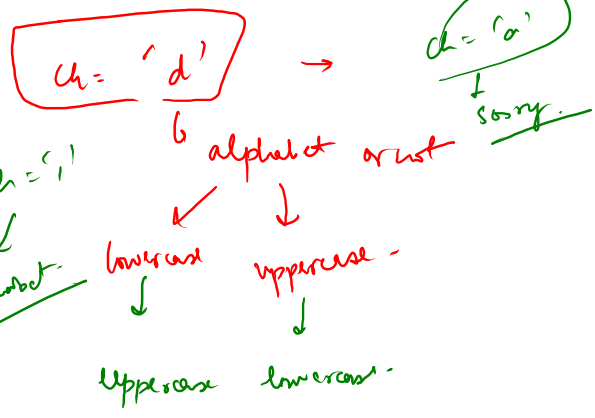
Take in a character as an input from the user, A. Condition 1 : If the character is an alphabet then you need to toggle the character first, For example, if the entered character is 'a', then convert it into 'A', and if the entered character is 'A' then convert it into 'a', this simply means that if the entered character is a capital case then convert it into a small case character and vice-versa. After toggling the character, a. if the resultant character is not 'a', 'A', 'b', 'B', then take two jumps to the left and print the character, for eg. If the toggled character is 'c' then print 'a', If the toggled character is 'Z', then print 'X'. b. If the toggled character is 'a', 'A', 'b', 'B', then print "Sorry". B. Condition 2: Otherwise print "No alphabet"

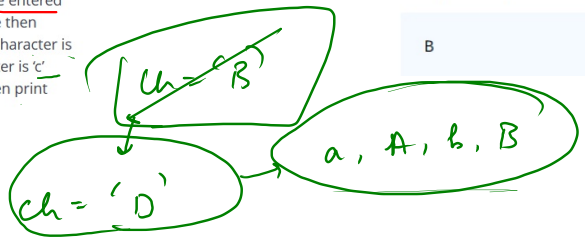
Sample Input 0

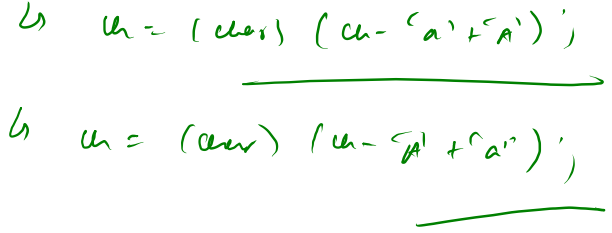
```
d
```

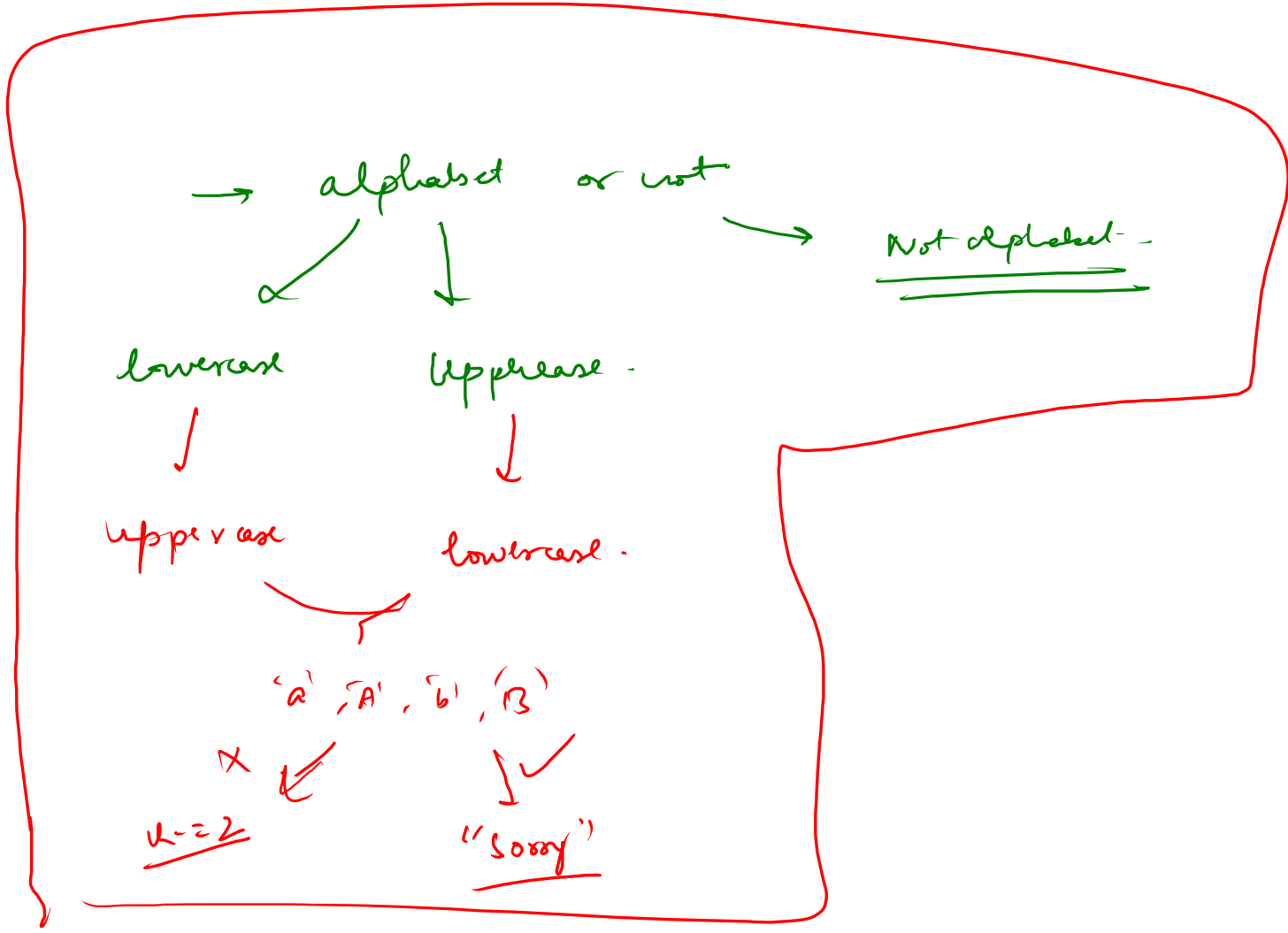
Sample Output 0

```
B
```





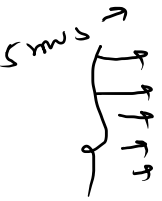


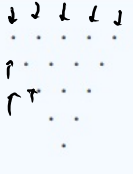

alphabet or not
Not alphabet
lowercase
Uppercase
uppercase
lowercase
'a' 'A' 'b' 'B'
k = 2
"Sorry"

```java
public class Solution {
    public static String toggleAndJump(char ch){
        if(ch >= 'a' && ch <= 'z'){
            ch = (char)(ch-'a'+'A');
        }else if(ch >='A' && ch <='Z'){
            ch = (char)(ch-'A'+'a');
        }else{
            return "No alphabet";
        }

        if(ch == 'a' || ch == 'A' || ch == 'b' || ch == 'B'){
            return "Sorry";
        }
        else {
            ch-=2;
            return ch +"";
        }

    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Yo
        Scanner scn = new Scanner(System.in);
        char ch = scn.next().charAt(0);

        System.out.println(toggleAndJump(ch));
    }
}
```

Print the given pattern1

Problem | Submissions | Leaderboard | Discussions





nsp
0
1
2
3
1

(dst)
ns +
5
4
3
2
1

```java
public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN.
        int nsp = 0;
        int nst = 5;

        for(int i=1;i<=5;i++){
            for(int j=1;j<=nsp;j++){
                System.out.print(" ");
            }

            for(int j=1;j<=nst;j++){
                System.out.print(". ");
            }

            nsp++;
            nst--;
            System.out.println();
        }
    }
}
```

Divide n by 2 3 5 and tell steps(18 Jun)

Problem | Submissions | Leaderboard | Discussions

Take a natural number n as an integer input, and variable steps of integer type as input. Then perform the following operations on it.

a. If the number is divisible by 2, then keep on dividing the number n by 2, till the time the number is divisible by 2 and also increment the variable steps by 2, each time you divide the number by 2.

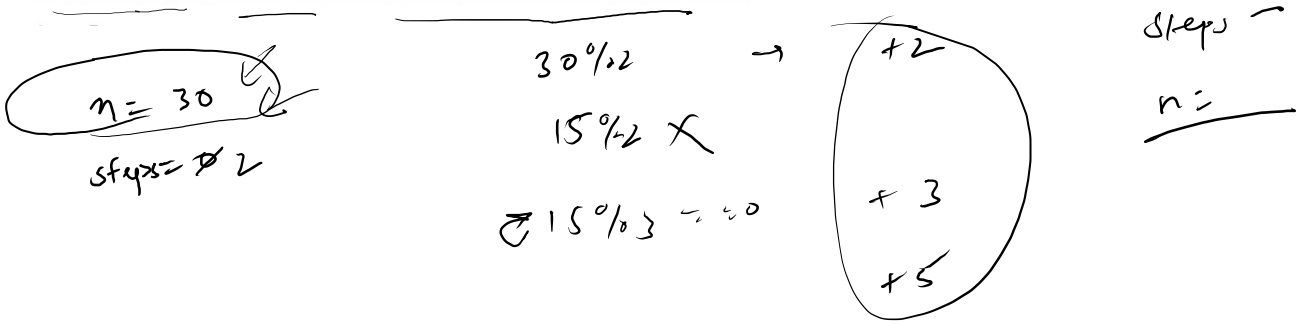
b. Also, check If the number is divisible by 3, then keep on dividing the number n by 3, till the time the number is divisible by 3 and also increment the variable steps by 3, each time you divide the number by 3. c. Also, If the number is divisible by 5, then keep on dividing the number n by 5, till the time the number is divisible by 5 and also increment the variable steps by 5, each time you divide the number by 5. In the end print the value of the variable steps in the first line and the final value of number n in the second line.

Sample Input 0

```
30
0
```

Sample Output 0

```
10
1
```



```
public static void divide(int n, int steps){
    while(n%2 == 0){
        steps+=2;
        n/=2;
    }

    while(n%3==0){
        steps += 3;
        n/=3;
    }

    while(n%5 == 0){
        steps += 5;
        n/=5;
    }

    System.out.println(steps);
    System.out.println(n);
}

public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print o
    Scanner scn = new Scanner(System.in);
    int n =scn.nextInt();
    int steps = scn.nextInt();

    divide(n,steps);
}
```

Second Largest in array(2 july)

Problem | Submissions | Leaderboard | Discussions

You have been given a random integer array/list(ARR) of size N. You are required to find and return the second largest element present in the array/list.

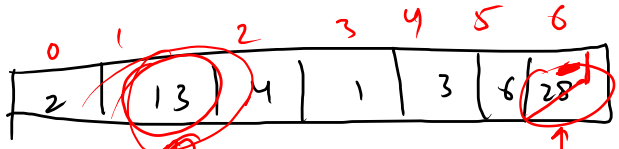
If N <= 1 or all the elements are same in the array/list then return -2147483648 or -2 ^ 31(It is the smallest value for the range of Integer)

Sample Input 0

```
7
2 13 4 1 3 6 28
```

Sample Output 0

```
13
```



n=7

→ sort →

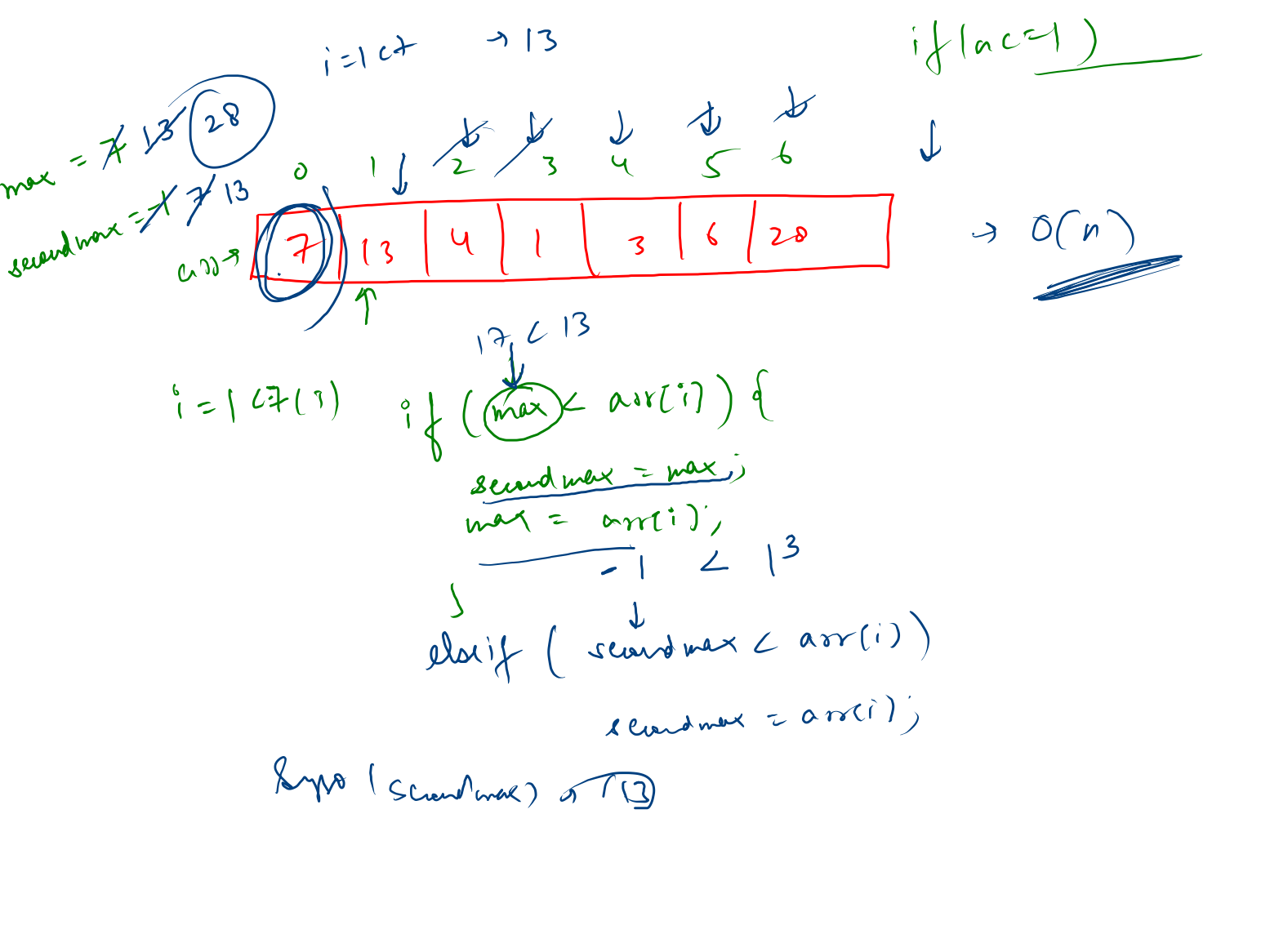
for(int i=1; i<=2; i++)

13

13

if(n<=1) → return -

O(n²)


i=1 ct → 13
if(ac=1)
max = 28
secondmax = 13
0 1 2 3 4 5 6
7 13 4 1 3 6 20
→ O(n)
17 < 13
i=1 ct(1)
if ((max < arr[i]) {
secondmax = max;
max = arr[i];
-1 < 13
elsif (secondmax < arr[i])
secondmax = arr[i];
Syso (secondmax) of 13

```java
public static int secondLargest(int arr[],int n){
    if(n<=1){
        return  -2147483648;
    }

    int max = -2147483648;
    int secondMax = -2147483648;

    for(int i=0;i<n;i++){
        if(max<arr[i]){
            secondMax = max;
            max = arr[i];
        }
        else if(secondMax < arr[i] && max != arr[i]){
            secondMax = arr[i];
        }
    }

    return secondMax;
}

public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output to ST
    Scanner scn = new Scanner(System.in);
    int n =scn.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<n;i++){
        arr[i] = scn.nextInt();
    }

    System.out.println(secondLargest(arr,n));
}
```

Declare the first array of size n that stores values of int data-type. Then take n integer inputs and store them in the array one by one. For each index print the sum of all the elements except the element present at that index..

Input Format

N as Size of Array then N Int value as Arr[i] values

Constraints

NA

Output Format

Print value of sum of array except that particular idx

Sample Input 0

```
4
2
7
8
9
```

Sample Output 0

```
24
19
18
17
```