1157. 단어공부

: 단어에서 가장 많이 사용된 알파벳 찾기

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		String s = br.readLine();
		s = s.toUpperCase();

		int alArr [] = new int[26];
		int max = 0;
		int maxIdx = '?';

		for(int i=0; i<s.length(); i++) {
			int al = s.charAt(i)-'A';
			alArr[al]++;
			if(alArr[al] > max) {
				max = alArr[al];
				maxIdx = al;
			}
			else if(alArr[al] == max) {
				maxIdx = '?' - 'A';
			}
		}
		char ans = (char)(maxIdx+'A');
		System.out.println(ans);
	}
}
```

1. 문자열을 모두 대문자로 변경
2. 알파벳 개수만큼의 크기를 가진 배열 생성
3. 알파벳 별 인덱스는 A:0 ~ Z:25 (문자에서 아스키코드 'A'만큼을 뺌)
4. 문자열을 순회하면서 알파벳별 등장 횟수를 배열에 저장함:
   1) 등장하는 알파벳의 인덱스에 +1을 해줌.
   2) 해당 알파벳이 이전 최댓값보다 크다면, 해당 값을 최댓값으로 하고 인덱스를 저장함.
   3) 만약 최댓값이 동일하다면 '?'를 저장함.

2941. 크로아티아 알파벳

: 크로아티아 알파벳의 개수 세기

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		String s = br.readLine();
		int cnt = 1;

		for(int i=1; i<s.length(); i++) {
			if(s.charAt(i) == '=' || s.charAt(i) == '-') {
				if(i!=1 && (s.charAt(i-1) == 'z' && s.charAt(i-2) == 'd'))
					cnt--;
			}
			else if(s.charAt(i) == 'j') {
				if(s.charAt(i-1) != 'l' && s.charAt(i-1) != 'n')
					cnt++;
			}
			else
				cnt++;
		}
		System.out.println(cnt);
	}
}

```

1. 끝 글자를 기준으로 크로아티아 알파벳 판별

   '=' : c=, s=, dz=, z=

   '-' : c-, d-

   'j' : lj, nj

2. 문자열을 순회하면서 (=, -, j) 이외의 문자를 만나면 단어개수 +1 함

3. 크로아티아알파벳의 (=, -, j)는 첫 문자에 나올 수 없으므로 두번째 문자부터 순회.

   1) '='인 경우: 이전 문자들이 dz 였다면 d 와 z 에서 개수를 총 두 번 셌으므로 개수 -1 함.

   2) '-'인 경우: 이미 이전 문자에서 개수를 셌으므로 세지 않고 넘어감.

   3) 'j'인 경우: 이전 문자가 l 이나 n 이 아니었다면 크로아티아 문자가 아니므로 문자 'j'를 count (개수+1)

2745. 진법변환

: B 진수 -> 10 진수 변환

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringTokenizer st = new StringTokenizer(br.readLine());
		String N = st.nextToken();
		int B = Integer.parseInt(st.nextToken());
		int num, dec = 0;
		int n = N.length()-1;
		for(char ch: N.toCharArray()) {
			if('A'<=ch && ch<='Z')
				num = ch - 55;
			else
				num = ch - '0';
			dec += Math.pow(B, n) * num;
			n--;
		}
		System.out.println(dec);
	}
}

```

10 진수 = B 진수^자릿수 * B 진수숫자의 합

1. B 진수 숫자를 문자열로 받아 총 자릿수를 구함. 0 승부터니까 길이 -1
2. 첫번째 문자부터 하나씩 접근.
3. 문자가 알파벳이면 숫자로 바꿈(아스키코드 -55)
4. 문자가 알파벳이 아니면 char 형을 숫자형으로 바꿔줌
5. 진수^자릿수 * 숫자 계산해 누적합한 후, 자릿수를 1 씩 줄이면서 문자열이 끝날 때까지 반복

11005. 진법변환 2

: 10 진수 -> B 진수 변환

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringTokenizer st = new StringTokenizer(br.readLine());
		StringBuilder sb = new StringBuilder();

		int N = Integer.parseInt(st.nextToken());
		int B = Integer.parseInt(st.nextToken());
		int rem = 0;

		while(N > 0) {
			rem = N%B;
			if(rem >= 10)
				sb.append((char)(rem + 55));
			else
				sb.append(rem);
			N/=B;
		}

		System.out.println(sb.reverse());
	}
}

```

몫이 0 이 될 때까지 N 을 B 로 나누고, 몫이 0 이 되면 나머지에 역순으로 접근.

1. 10 진수 N 이 0 보다 클 동안 반복:
   1) N 을 B 로 나눈 나머지를 저장
   2) 만약 나머지가 10 이상이면 알파벳으로 변경(아스키코드 +55)
   3) 구한 나머지를 StringBuilder 에 추가.
   4) 나눗셈 진행 (N / B)
   5) N 이 0 이 되면 반복문 종료됨.
2. 나머지를 역순으로 출력해야하므로 reverse()

1193. 분수찾기

: X 번째 분수 구하기

1/1	1/2	1/3	1/4	1/5	...
2/1	2/2	2/3	2/4	...	...
3/1	3/2	3/3	...	...	...
4/1	4/2	...	...	...	...
5/1	...	...	...	...	...
...	...	...	...	...	...

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
    public static void main(String [] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringBuilder sb = new StringBuilder();
        int X = Integer.parseInt(br.readLine());
        int n = 1;
        int denum, num;
        while(true) {
                if(X <= n*(n+1)/2)
                        break;
                n++;
        }
        if(n%2==1) {
                denum = X - n*(n-1)/2;
                num = n-denum+1;
        }
        else {
                num = X - n*(n-1)/2;
                denum = n-num+1;
        }
        sb.append(num+"/"+denum);
        System.out.println(sb);
    }
}

```

규칙:

n:	1	2		3			4				5				
분모:	1	2	1	1	2	3	4	3	2	1	1	2	3	4	5
분자:	1	1	2	3	2	1	1	2	3	4	5	4	3	2	1
X:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

대각선 1 개 = 라인 1 개.

라인=n 이라고 하면, 라인별로 분수를 n 개씩 가지고 있음.

분모 -> 홀수라인: 1 부터 n 까지 1 씩 증가 / 짝수라인: n 부터 1 까지 1 씩 감소

분자 -> 홀수라인: n 부터 1 까지 1 씩 감소 / 짝수라인: 1 부터 n 까지 1 씩 증가

1. 분수가 몇번라인에 있는지 찾기.
    1) n 번 라인의 마지막 인덱스 = 1 부터 n 까지의 합 (n*(n+1)/2)
    2) n 을 하나씩 증가시키면서 위치 찾기.
        - X 가 n 번 라인의 마지막 인덱스보다 크다면 다음 라인들 중에 있다는 의미, 작거나 같다면 n 번 라인에 있다는 의미.
2. 홀수라인일 경우:
    1) 분모: X 에서 이전라인의 마지막 인덱스를 뺀 것이 분모. (분모가 1 부터 증가하는 형태이므로 라인 n 의 첫번째 인덱스를 1 로 만들어줌)
    2) 분자: n 부터 1 까지 감소하는 형태. 분모가 1 부터 n 까지 증가하는 형태였으므로, n 에서 분모만큼을 뺀다. 이 값은 실제 분자 값보다 작으므로 1 을 더해준다.

1018. 체스판 다시 칠하기

: 8x8 체스판에서 다시칠해야 하는 칸의 최소개수 구하기

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.StringTokenizer;


public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringTokenizer st = new StringTokenizer(br.readLine());
		int N = Integer.parseInt(st.nextToken());
		int M = Integer.parseInt(st.nextToken());

		int chess [][] = new int [N][M];
		int cnt = 0;
		int min = 65;

		for(int n=0; n<N; n++) {
			String line = br.readLine();
			for(int m=0; m<M; m++)
				chess[n][m] = line.charAt(m);
		}

		for(int i=0; i<=N-8; i++) {
			for(int j=0; j<=M-8; j++) {

				for(int n=0; n<8; n++) {
					for(int m=0; m<8; m++) {
						if((n%2==0 && m%2==0) || (n%2==1 && m%2==1)){
							if(chess[n+i][m+j] != 'W')
								cnt++;
						}
						else {
							if(chess[n+i][m+j] != 'B')
								cnt++;
						}
					}
				}
				cnt = Math.min(cnt, 64-cnt);
				min = Math.min(min, cnt);
				cnt=0;
			}
		}

		System.out.println(min);
	}
}

```

규칙:

* 첫 칸이 W: 짝수행-짝수열, 홀수행-홀수열이 W. (인덱스가 0 부터 시작했을 때)

* 첫 칸이 B 일 때의 바꾼 개수 = (전체 64) – (첫 칸이 W 일 때의 바꾼 개수)

1. 8x8 만큼씩 돌면서 바꾼 개수 세기
2. 첫 칸이 W 인 체스판으로 바꾼다고 가정.
3. 첫 칸이 W 인 경우: (짝수행-짝수열), (홀수행-홀수열)이 W 여야 함. 만약 W 가 아니면 바꿔야 하므로 개수+1
4. 반대의 경우에 B 가 아니면 개수+1
5. 첫 칸이 B 일 때의 바꾼 개수 = (전체 64) – (첫 칸이 W 일 때의 바꾼 개수)
6. 바꾼 개수는 첫 칸이 B 일 때와 W 일 때 중 최솟값으로.

2750. 수 정렬하기

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
	public static void SelectSort(int k, int [] arr) {
		int min = 1001;
		int minIdx = 0;
		int tmp;
		for(int i=k; i<arr.length; i++) {
			if(min>arr[i]) {
				min = arr[i];
				minIdx = i;
			}
		}
		tmp = arr[minIdx];
		arr[minIdx] = arr[k];
		arr[k] = tmp;
	}
	public static void main (String [] args) throws Exception{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringBuilder sb = new StringBuilder();
		int N = Integer.parseInt(br.readLine());
		int arr [] = new int[N];

		for(int i=0; i<N; i++)
			arr[i] = Integer.parseInt(br.readLine());

		for(int i=0; i<arr.length; i++) {
			SelectSort(i,arr);
			sb.append(arr[i]+"\n");
		}
		System.out.println(sb);
	}
}

```

선택정렬 이용

1. i = 0 부터 배열 끝까지 순차탐색
2. i 이후의 최솟값과 i 번째 원소를 swap

2751. 수 정렬하기 2

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
	public static int [] sortedArr;

	public static void Merge(int [] arr, int start, int middle, int end) {
		int left = start;
		int right = middle+1;
		int now = start;

		while(left<=middle && right<=end) {
			if(arr[left] > arr[right])
				sortedArr[now++] = arr[right++];

			else
				sortedArr[now++] = arr[left++];

		}

		if(left>middle) {
			while(right<=end)
				sortedArr[now++] = arr[right++];
		}
		else {
			while(left<=middle)
				sortedArr[now++] = arr[left++];
		}

		for(int i=start; i<=end; i++) {
			arr[i] = sortedArr[i];
		}
	}

	public static void MergeSort(int [] arr, int start, int end) {
		if(start < end) {
			int middle = (start + end) / 2;
			MergeSort(arr, start, middle);
			MergeSort(arr, middle+1, end);
			Merge(arr,start,middle,end);
		}
	}

	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringBuilder sb = new StringBuilder();
		int N = Integer.parseInt(br.readLine());
		int [] arr = new int[N];
		sortedArr = new int[N];

		for(int i=0; i<N; i++)
			arr[i] = Integer.parseInt(br.readLine());

		MergeSort(arr,0,N-1);

		for(int i=0; i<N; i++)
			sb.append(arr[i]).append("\n");
		System.out.println(sb);
	}
}

```

합병정렬 이용: 분할 후 합병정렬

1. 분할: 배열의 원소 개수가 1 개가 될 때까지 배열을 1/2 로 분할함.
2. 합병: 분할한 두 배열의 원소를 차례로 비교하고, 별도의 배열에 합병 정렬
    1) 두 배열의 원소 중 더 작은 원소를 sortedArr 에 먼저 저장.

2) 두 배열 중 한 배열이라도 모두정렬되면 반복문 종료.

3) 반복문 종료 후 남은 하나의 배열을 그대로 sortedArr 에 저장

10989. 수 정렬하기 3

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringBuilder sb = new StringBuilder();
		int N = Integer.parseInt(br.readLine());
		int sorted [] = new int[10001];

		for(int i=0; i<N; i++) {
			int n = Integer.parseInt(br.readLine());
			sorted[n]++;
		}
		for(int i=0; i<sorted.length; i++) {
			while(sorted[i] > 0) {
				sb.append(i).append("\n");
				sorted[i]--;
			}
		}
		System.out.println(sb);
	}
}

```

계수정렬 이용: 입력범위만큼의 배열을 만들고 배열에 숫자 개수를 누적

1. 입력범위만큼의 크기를 가지는 배열 생성
2. 입력받은 숫자의 인덱스에 숫자 등장 개수를 누적합함.
3. i=0 부터 배열값이 0 이 아닌 인덱스번호를 순차적으로 출력.
4. i 번째 원소값을 -1 씩 감소시키면서, 값이 0 이 될때까지 반복출력함

11650. 좌표 정렬하기

1. x 좌표 오름차순 2. x 좌표 같으면 y 좌표 오름차순 정렬

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.StringTokenizer;

public class Main {
	public static void main(String [] args) throws Exception {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		StringBuilder sb = new StringBuilder();
		int N = Integer.parseInt(br.readLine());
		int [][] point = new int[N][2];

		for(int i=0; i<N; i++) {
			StringTokenizer st = new StringTokenizer(br.readLine());
			point[i][0] = Integer.parseInt(st.nextToken());
			point[i][1] = Integer.parseInt(st.nextToken());
		}

		Arrays.sort(point, (p1,p2)->{
			if(p1[0]==p2[0])
				return p1[1]-p2[1];
			else
				return p1[0]-p2[0];
		});

		for(int i=0; i<N; i++) {
			sb.append(point[i][0]).append(" ")
			.append(point[i][1]).append("\n");
		}
		System.out.println(sb);
	}
}

```

1. Arrays.sort 를 사용자정의해서 정렬
2. Arrays.sort → 반환값이 양수: a1 > a2 / 음수: a1 < a2 / 0: a1 == a2

   → a1 – a2 를 통해 대소비교 가능.
3. x 좌표가 같으면 y 좌표끼리 대소비교한 결과 반환
4. x 좌표가 다르면 x 좌표끼리만 대소비교한 결과 반환