1) 문자열 (문제)

1-1) 문자열 (문법)

---

3. 문장 속 가장 긴 단어 찾기 (단어 길이 세기)

: 3_longest_word_in_sentence.java

3. 문장 속 단어

설명

한 개의 문장이 주어지면 그 문장 속에서 가장 긴 단어를 출력하는 프로그램을 작성하세요.

문장속의 각 단어는 공백으로 구분됩니다.

입력

첫 줄에 길이가 100을 넘지 않는 한 개의 문장이 주어집니다. 문장은 영어 알파벳으로만 구성되어 있습니다.

출력

첫 줄에 가장 긴 단어를 출력한다. 가장 길이가 긴 단어가 여러개일 경우 문장속에서 가장 앞쪽에 위치한

단어를 답으로 합니다.

예시 입력 1

```
it is time to study
```

예시 출력 1

```
study
```

---

4. 단어 거꾸로 뒤집기

: 4_flip_word.java

4. 단어 뒤집기

설명

N개의 단어가 주어지면 각 단어를 뒤집어 출력하는 프로그램을 작성하세요.

입력

첫 줄에 자연수 N(3<=N<=20)이 주어집니다.

두 번째 줄부터 N개의 단어가 각 줄에 하나씩 주어집니다. 단어는 영어 알파벳으로만 구성되어 있습니다.

출력

N개의 단어를 입력된 순서대로 한 줄에 하나씩 뒤집어서 출력합니다.

예시 입력 1

```
3
good
Time
Big
```

예시 출력 1

```
doog
emiT
giB
```

1. **String → char array로 변환**

   : **char[] arr = str.toCharArray(); - String → char array 변환**

```
char[] arr = str.toCharArray();
```

2. **char array에서 거꾸로 char 1개씩 뽑기 → answer에 추가**

```
String answer = ""

for(int i=str.length-1; i>=0; i--){
    answer += s[i];
}
```

- return 값이 ArrayList일때, **String → ArrayList 변환**

  : **answer.add(tmp); - ArrayList에 String값 넣기**

```
String tmp = "";

for(int i=str.length-1; i>=0; i--){
```

```
    tmp += s[i];   // 1. tmp에 char 1개 저장
}

answer.add(tmp); // 2. string → ArrayList에 저장
```

- **char array 전체 → String 변환**

  : **String.valueOf(char_arr);**

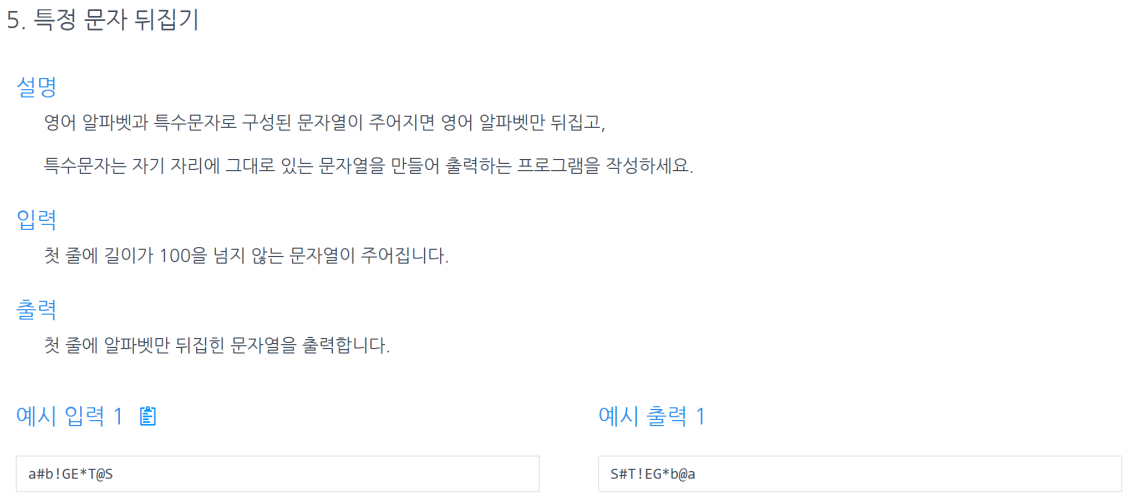
```
String answer = "";

answer = String.valueOf(char_arr); // 전체 char array를 String으로 변환
```

---

5. 특수문자 빼고 char만 뒤집기

: 5_flip_word_not_special_word.java



5. 특정 문자 뒤집기

설명

영어 알파벳과 특수문자로 구성된 문자열이 주어지면 영어 알파벳만 뒤집고,

특수문자는 자기 자리에 그대로 있는 문자열을 만들어 출력하는 프로그램을 작성하세요.

입력

첫 줄에 길이가 100을 넘지 않는 문자열이 주어집니다.

출력

첫 줄에 알파벳만 뒤집힌 문자열을 출력합니다.

예시 입력 1

a#b!GE*T@S

예시 출력 1

S#T!EG*b@a

1. **String → char array로 변환**

```
char[] arr = str.toCharArray();
```

2. **left, right 인덱스 번호 설정**

```
int left = 0;
int right = str.length()-1;
```

3. **char array 돌면서, left랑 right 값이 알파벳인지 각각 비교**

   : **Character.isAlphabetic(arr[i]); - 알파벳인지 확인하는 함수**

   → left == 특수문자 일때 : 그냥 다음으로 넘어가기

   → right == 특수문자 일때 : 그냥 왼쪽으로 넘어가기

   → 둘다 아닐때 (left, right 둘다 char일때) : left right SWAP

```
while(left < right){
    if(!Character.isAlphabetic(arr[left])) left++;
    else if(!Character.isAlphabetic(arr[right])) right--;
    else{
        char tmp = arr[left];
        arr[left] = arr[right];
        arr[right] = tmp;

        left++;
        right--;
    }
}
```

4. **char array → String 변환**

   : **answer = String.valueOf(arr); - char array → String 변환**

```
answer = String.valueOf(arr);
```

---

6. 중복 문자 제거 (char가 처음으로 등장하는 위치 확인)

: 6_remove_duplicated_char.java

6. 중복문자제거

설명

소문자로 된 한개의 문자열이 입력되면 중복된 문자를 제거하고 출력하는 프로그램을 작성하세요.

중복이 제거된 문자열의 각 문자는 원래 문자열의 순서를 유지합니다.

입력

첫 줄에 문자열이 입력됩니다. 문자열의 길이는 100을 넘지 않는다.

출력

첫 줄에 중복문자가 제거된 문자열을 출력합니다.

예시 입력 1	예시 출력 1
ksekkset	kset

- **처음으로 등장하는 char만 저장**

  : **str.indexOf(str.charAt(i)) - 처음으로 등장하는 char의 index 반환**

```
// [현재 index == 처음으로 발견된 index] : 이 경우를 제외하고 나머지 문자들을 삭제!
for(int i=0; i<str.length; i++){
    if(i == str.indexOf(str.charAt(i)))   answer += str.charAt(i); // i번째 char를 저장
}
```

---

7. 회문 문자열 (Palindrome)

: 7_palindrome.java

7. 회문 문자열

설명

앞에서 읽을 때나 뒤에서 읽을 때나 같은 문자열을 회문 문자열이라고 합니다.

문자열이 입력되면 해당 문자열이 회문 문자열이면 "YES", 회문 문자열이 아니면 "NO"를 출력하는 프로그램을 작성하세요.

단 회문을 검사할 때 대소문자를 구분하지 않습니다.

입력

첫 줄에 길이 100을 넘지 않는 공백이 없는 문자열이 주어집니다.

출력

첫 번째 줄에 회문 문자열인지의 결과를 YES 또는 NO로 출력합니다.

예시 입력 1	예시 출력 1
gooG	YES

1. **단순 비교 (left, right)**

   : 전체 str 대문자화 → left, right 비교 → (전체 문자열 길이 / 2) 직전까지 비교

```
str = str.toUpperCase();

int left = 0;
int right = str.length()-1;

for(int i=0; i<str.length()/2; i++){
    if(str.charAt(left) == str.charAt(right)){
        left++;
        right--;
        answer = "YES";
    }
    else{
        answer = "NO";
        break;
    }
}
```

2. **StringBuilder 사용**

   : string 뒤집어서 둘이 equal한지 비교

: **String tmp = new StringBuilder(str).reverse().toString(); - StringBuilder로 뒤집기**

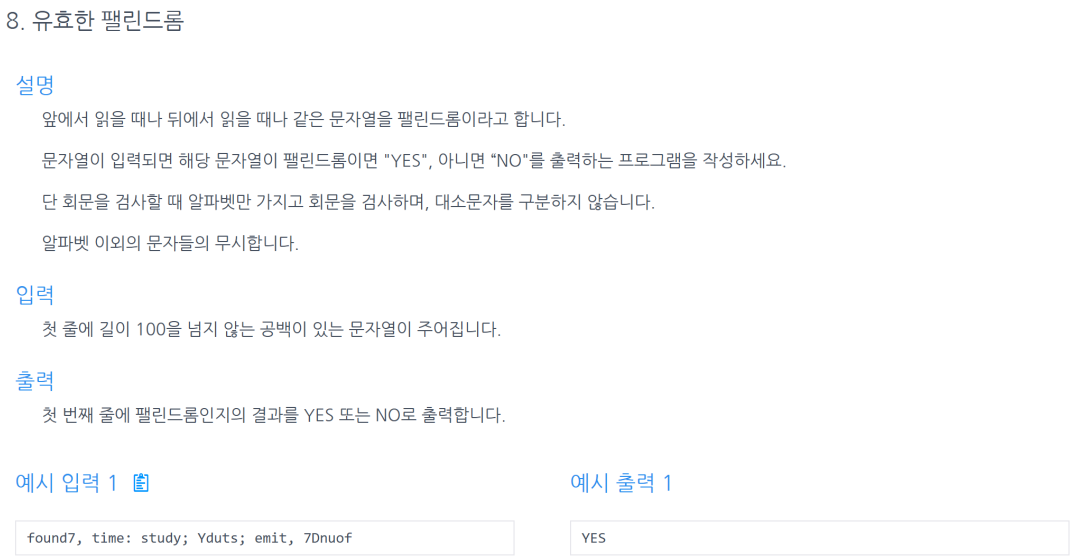
```
String answer = "NO";

String tmp = new StringBuilder(str).reverse().toString(); // 1. str 뒤집기
if(str.equalsignoreCase(tmp)) answer = "YES";

return answer;
```

3. **문자 여러개 + 숫자 + 특수문자 + 공백 포함 → 알파벳만 검사**

   ex) found7, time: study; Yduts; emit, 7Dnuof → “YES”



8. 유효한 팰린드롬

설명

앞에서 읽을 때나 뒤에서 읽을 때나 같은 문자열을 팰린드롬이라고 합니다.

문자열이 입력되면 해당 문자열이 팰린드롬이면 "YES", 아니면 "NO"를 출력하는 프로그램을 작성하세요.

단 회문을 검사할 때 알파벳만 가지고 회문을 검사하며, 대소문자를 구분하지 않습니다.

알파벳 이외의 문자들의 무시합니다.

입력

첫 줄에 길이 100을 넘지 않는 공백이 있는 문자열이 주어집니다.

출력

첫 번째 줄에 팰린드롬인지의 결과를 YES 또는 NO로 출력합니다.

예시 입력 1

```
found7, time: study; Yduts; emit, 7Dnuof
```

예시 출력 1

```
YES
```

- **해결 : replaceAll(”[A-Z]”, “”);**

1) 문자열 전부 대문자화

2) 알파벳 A~Z까지가 아닌 것들 → 전부 삭제해버리기

```
str = str.toUpperCase().replaceAll("[^A-Z]", "");
```

3) StringBuilder 사용 : reverse 한 뒤, 둘이 같은지 비교

```
String tmp = new StringBuilder(str).reverse().toString();

if(str.equals(tmp)) answer = "YES";
```

😈 반드시 'nextLine()' 으로 입력받을 것!

(한 줄 전체를 입력받아야 함 - 공백, 특수문자 때문에 입력 멈추는 일 X)

---

8. 문자, 숫자 섞여있는 문자열 중, 숫자만 추출

: 8_find_number_only.java

9. 숫자만 추출

설명

문자와 숫자가 섞여있는 문자열이 주어지면 그 중 숫자만 추출하여 그 순서대로 자연수를 만듭니다.

만약 "tge0a1h205er"에서 숫자만 추출하면 0, 1, 2, 0, 5이고 이것을 자연수를 만들면 1205이 됩니다.

추출하여 만들어지는 자연수는 100,000,000을 넘지 않습니다.

입력

첫 줄에 숫자가 섞인 문자열이 주어집니다. 문자열의 길이는 100을 넘지 않습니다.

출력

첫 줄에 자연수를 출력합니다.

예시 입력 1	예시 출력 1
g0en2T0s8eSoft	208

1. ASCII 값으로 비교 (48 ~ 57 = 숫자 0 ~ 9)

1. **String → char array로 바꾼뒤, 문자 1개씩 비교**

```
for(char x : str.toCharArray()){
    ~~
}
```

2. **ASCII 값으로 숫자인지 확인**
   - 48 ~ 57 : 숫자 (0 ~ 9)

```
if(x >= 48 && x <= 57) // 숫자 0 ~ 9
```

3. **answer에 10씩 곱해가면서 숫자 더하기**

   : answer = answer * 10 + (x - 48)

   → 기존 answer값이 윗자리수로 이동, (x - 48)은 현재 숫자값

```
answer = answer * 10 + (x - 48)
```

ex) 문자열 속 숫자가 (1, 2, 0, 5), 초기 answer = 0

→ 첫 번째 : answer = 0 * 10 + 1 = 1

→ 두 번째 : answer = 1 * 10 + 2 = 12

→ 세 번째 : answer = 12 * 10 + 0 = 120

→ 네 번째 : answer = 120 * 10 + 5 = 1205

∴ 1205 완성

2. char 1개씩 숫자인지 확인하기

1. **해당 char가 숫자인지 확인**

   : **Character.isDigit(x) - 숫자인지 확인**

```
String answer = "";

for(char x : str.toCharArray()){
    if(Character.isDigit(x)) answer += x;
}
```

2. **string 값을 int 값으로 바꿔서 return**

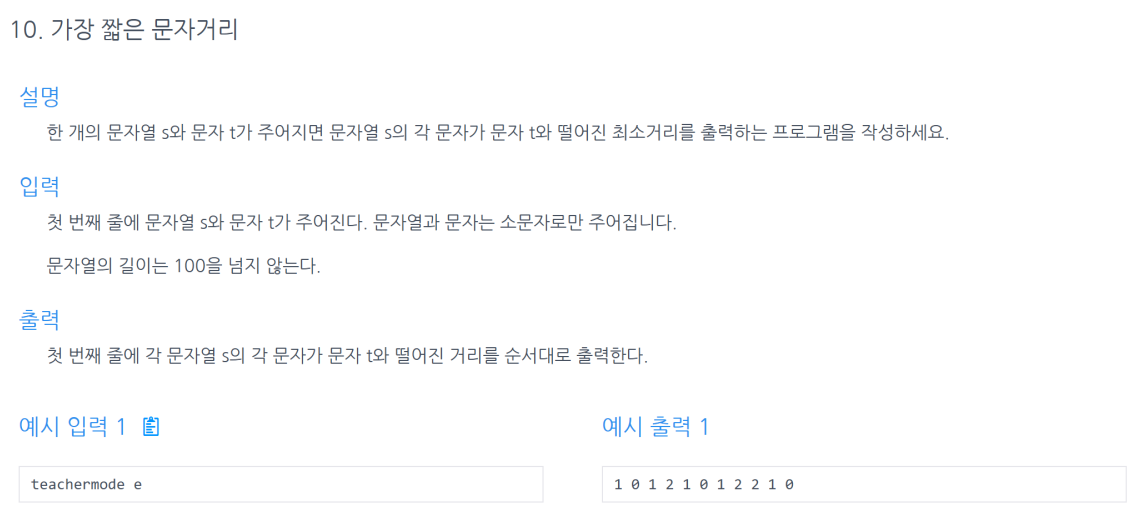
   : **Integer.parseInt(answer); - String → int 변환**

return Interger.parseInt(answer); // 이러면 String 값이 Int 값으로 바뀜

---

9. 문자열, 문자 간 가장 짧은 거리 출력하기

: 9_shortest_char_distance



10. 가장 짧은 문자거리

설명

한 개의 문자열 s와 문자 t가 주어지면 문자열 s의 각 문자가 문자 t와 떨어진 최소거리를 출력하는 프로그램을 작성하세요.

입력

첫 번째 줄에 문자열 s와 문자 t가 주어진다. 문자열과 문자는 소문자로만 주어집니다.

문자열의 길이는 100을 넘지 않는다.

출력

첫 번째 줄에 각 문자열 s의 각 문자가 문자 t와 떨어진 거리를 순서대로 출력한다.

예시 입력 1

```
teachermode e
```

예시 출력 1

```
1 0 1 2 1 0 1 2 2 1 0
```

ex) teachermode e

: 처음에 p = 1000으로 대충 잡고 시작 (p = e와의 거리 차이)

∴ e가 아니면, p를 1 증가시키기

[한 바퀴 : 왼쪽의 e로부터의 거리 측정] : 왼 → 오

→ t : 왼쪽에 e 없음 - **p=1000 & 1000 대입**

→ e : 자기 자신 - **p=0 & 0 대입**

→ a : e 아님 - **p 증가 (p=1) & 1 대입**

→ c : e 아님 - **p 증가 (p=2) & 2 대입**

→ h : e 아님 - **p 증가 (p=3) & 3 대입**

→ e : 자기 자신 - **p=0 초기화 & 0 대입**

...

→ o : e 아님 - **p 증가 (p=3) & 3 대입**

→ d : e 아님 - **p 증가 (p=4) & 4 대입**

→ e : 자기 자신 - **p=0 & 0 대입**

[두 바퀴 : 오른쪽의 e로부터의 거리 측정 → 값 더 작으면 갱신!] : 오 → 왼

→ e : 자기 자신 - **p=0 & 0 대입**

→ d : e 아님 - **p=1**

⇛ 기존 (p=4) 보다 작음

∴ p=1 로 갱신

→ o : e 아님 - p=2

⇛ 기존 (p=3) 보다 작음

∴ p=2 로 갱신

→ m : e 아님 - p=3

⇛ 기존 (p-2) 보다 큼

∴ 갱신 X

…

∴ 앞뒤로 2바퀴 돌면, int 배열 완성!

1. '문자열 + 문자 1개' 이어서 받기

```
public static void main(String args[]){
    Scanner sc = new Scanner(System.in);
    Main T = new Main();

    String str = sc.next();          // 띄어쓰기 전 (문자열) 저장
    char c = sc.next().charAt(0);    // 띄어쓰기 후 (문자 1개) 저장

}
```

2. p 생성, String을 charArray로 변환

```
int p = 1000;
char[] arr = str.toCharArray();
```

3. 한 바퀴 (왼→오) : charArray랑 char 1개씩 비교

1) 둘이 같음 : p=0

2) 둘이 다름 : ++p

```
// 2. [한 바퀴]
// charArray랑 char를 1개씩 비교하기
for(int i=0; i<str.length(); i++){
    if(arr[i] == c){
        p = 0;
        answer[i] = p;
    }
    else{
        ++p;
        answer[i] = p;
    }
}
```

4. 두 바퀴 (오→왼) : charArray랑 char 1개씩 비교

1) 둘이 같음 : p=0

2) 기존 answer[i] 보다 p가 작을 때 : 지금 p 값으로 갱신

3) 기존 answer[i] 보다 p가 클 때 : 그냥 넘어감

```
// 3. [두 바퀴]
// 반대 방향으로 char를 1개씩 비교하기 -> 더 작으면 값 갱신하기
for(int i=str.length()-1; i>=0; i--){
    if(arr[i] == c){
        p = 0;
        answer[i] = p;
```

```
    }
    else if(answer[i] > ++p){    // 기존 거리가 더 큼 -> 새로운 값으로 갱신
        answer[i] = p;
    }
    else{                        // 기존 거리가 더 작음 -> 갱신 X
        ++p;
    }
}
```

5. answer 배열 전달 → 배열 속 1개씩 출력

```
return answer;

...

for(int x : T.solution(str, c)){ // 반환되는 int 배열을 하나씩 출력
    System.out.print(x + " ");
}
```

- **전체 코드**

```
import java.util.Scanner;

public class Main {
    public int[] solution(String str, char c){
        int[] answer = new int[str.length()];

        // 1. p 생성, string을 charArray로 변환
        int p = 1000;
        char[] arr = str.toCharArray();

        // 2. [한 바퀴]
        // charArray랑 char를 1개씩 비교하기
        for(int i=0; i<str.length()-1; i++){
            if(arr[i] == c){
                p = 0;
```

```
            answer[i] = p;
        }
        else{
            ++p;
            answer[i] = p;
        }
    }

    // 3. [두 바퀴]
    // 반대 방향으로 char를 1개씩 비교하기 -> 더 작으면 값 갱신하기
    for(int i=str.length()-1; i>=0; i--){
        if(arr[i] == c){
            p = 0;
            answer[i] = p;
        }
        else if(answer[i] > ++p){    // 기존 거리가 더 큼 -> 새로운 값으로 갱신
            answer[i] = p;
        }
        else{                    // 기존 거리가 더 작음 -> 갱신 X
            ++p;
        }
    }

    return answer;
}

public static void main(String args[]){
    Scanner sc = new Scanner(System.in);
    Main T = new Main();

    String str = sc.next();         // 띄어쓰기 전 (문자열) 저장
    char c = sc.next().charAt(0);    // 띄어쓰기 후 (문자 1개) 저장

    for(int x : T.solution(str, c)){ // 반환되는 int 배열을 하나씩 출력
        System.out.print(x + " ");
    }
}
}
```

10. 문자열 속, 연속되는 같은 문자 개수 세서 집어넣기

: 10_string_continue_same.java

11. 문자열 압축

설명

알파벳 대문자로 이루어진 문자열을 입력받아 같은 문자가 연속으로 반복되는 경우 반복되는

문자 바로 오른쪽에 반복 횟수를 표기하는 방법으로 문자열을 압축하는 프로그램을 작성하시오.

단 반복횟수가 1인 경우 생략합니다.

입력

첫 줄에 문자열이 주어진다. 문자열의 길이는 100을 넘지 않는다.

출력

첫 줄에 압축된 문자열을 출력한다.

예시 입력 1	예시 출력 1
KKHSSSSSSSE	K2HS7E

예시 입력 2	예시 출력 2
KSTTTSEEKFKKKDJJGG	KST3SE2KFK3DJ2G2

- **(현재 문자, 다음 문자) 비교**

  → **(만약 둘이 같음)** : count++

  → **(만약 둘이 다름) & (count가 1보다 큼)** : 현재 알파벳, count를 answer에 넣기

  → **(그 외)** : 현재 문자를 answer에 넣기

1. 입력받은 String의 맨 뒤에, 빈칸 1개 추가하기

∵ 맨 마지막 문자는, 비교할 다음 문자가 없음

→ 맨 뒤에 빈칸 1개 추가해서 비교할 다음 문자를 만들기!

```
public String solution(String str){
    String answer = "";

    // 1. str의 뒤에 빈 문자 1개 추가 (나중에 맨 마지막 요소가 비교할 때, 빈 문자링
```

```
        str += " ";

        return answer;
    }
```

2. (현재 문자 == 다음 문자) 비교

: 초기 설정 - count = 1

→ **(만약 둘이 같음)** : count++

→ **(만약 둘이 다름) & (count가 1이 아님)** : 현재 알파벳, count를 순서대로 answer에 넣기

→ **(그 외)** : 현재 문자를 answer에 넣기

```
public String solution(String str){
    String answer = "";

    str += " ";

    // 2. (현재 문자, 다음 문자) 비교 -> 같으면 count 증가, 다르면 알파벳이랑 count
    int count = 1;

    for(int i=0; i<str.length()-1; i++){
        if(str.charAt(i) == str.charAt(i+1)){                    // 1. (현재 문자 == 다음
            count++;
        }
        else if((str.charAt(i) != str.charAt(i+1)) && (count != 1)){    // 2. (현재 문자
            answer += String.valueOf(str.charAt(i));                // 2-1. 현재 알파벳
            answer += String.valueOf(count);                // 2-2. 현재 count값 넣기
            count = 1;                                      // 2-3. count = 1로 초기화
        }                                                   // ex) K K H → K2
        else{                                               // 3. 그 외 경우
            answer += String.valueOf(str.charAt(i));
        }
    }

    return answer;
}
```

11.