동물 단어

동물들은 8비트 문자로 대화를 한다. 그리고 동물들의 단어길이는 반드시 6글자이다.

예를들면 아래와 같다.

10101010 11100011 10010001 10110111 10110101 00111101 - 단어 1

01101010 10100011 11110001 10110111 11010101 01011101 - 단어 2

동물언어를 연구하는 과학자는 단어를 확실하게 찾아냈다. 그래서 동물들이 대화하며 주고받는 단어를 받아서 해석해주는 의뢰를 받았다. 하지만 받은 자료에는 노이즈가 조금씩 존재한다.

노이즈의 특징은 아래와 같다.

1. 노이즈는 6글자중 1개이상 꼭 존재한다.
2. 만약 어떤 글자에 노이즈가 존재한다면 1비트만 노이즈가 생긴다.
3. 노이즈는 1 → 0 이거나 0 → 1 이 된다.

노이즈의 특징을 분석하여 원래의 단어를 찾는 프로그램을 개발하자.

아래와 같이 구현을 요구하였다.

함수명	기능	조건
void init(char c[], int cnt)	과학자가 확실하게 찾은 동물 문자들이 주어진다. 8비트짜리 cnt 개 주어진다.	3 <= cnt <=32
void word(char c[][6], int cnt)	동물단어들이 주어진다. 동물단어는 항상 6문자로 되어있다.	1 <= cnt < 20000
void find(char in[6], char out[6])	노이즈가 발생한 단어가 in으로 주어진다. in 을 분석하여 원래의 단어를 out에 입력한다.	조건이 해당하는 단어가 여러가지인 경우 어떤 단어라도 무관하다.

조건 : find 호출횟수는 10보다 크고 20000보다 작으며 50개의 TC 5초이내 수행

```
#define TC_COUNT 50
#define CHARACTER_MAX 32
#include <stdio.h>
#include <cstdlib>
extern void init(char c[], int cnt);
extern void word(char c[][6], int cnt);
extern void find(char in[6], char out[6]);
int characterCnt;
char characterList[CHARACTER_MAX];
int wordCnt;
char wordList[20000][6];
void createAnimalCharacter() {
	characterCnt = rand() % 30 + 3;
	for (register int i = 0; i < characterCnt; i++) {
		characterList[i] = rand() % 256 - 128;
	}
}
void createAnimalWord() {
	wordCnt = rand() % 20000 + 1;
	for (register int i = 0; i < wordCnt; i++) {
		for (register int j = 0; j < 6; j++) {
			int index = rand() % characterCnt;
			wordList[i][j] = characterList[index];
		}
	}
}
int check(char in[6], char out[6]) {
	int cnt = 0;
	for (int i = 0; i < 6; i++) {
		char c = in[i] ^ out[i];
		if (c == 0) {
			continue;
		}
		else if (c == 1 || c == 2 || c == 4 || c == 8 || c == 16 || c == 32 || c ==
			64 || c == -128) {
			cnt++;
		}
		else {
			return 0;
		}
	}
	if (cnt == 0) {
		return 0;
	}
	for (int i = 0; i < wordCnt; i++) {
		int cnt = 0;
		char* selectWord = wordList[i];
		int success = 1;
		for (int j = 0; j < 6; j++) {
			if (selectWord[j] == out[j]) {
			}
			else {
				success = 0;
				break;
			}
```

```
                }
                if (success == 1) {
                        return 1;
                }

        }
        return 0;
}
int findCall() {
        int findCnt = rand() % 19991 + 10;
        int score = 2;
        for (register int i = 0; i < findCnt; i++) {
                int wordIndex = rand() % wordCnt;
                char word[6];
                char selectCharacter = 0;
                while (1) {
                        for (int j = 0; j < 6; j++) {
                                char noiseBool = rand() % 2;
                                selectCharacter |= noiseBool;
                                selectCharacter = selectCharacter << 1;
                        }
                        if (selectCharacter != 0) break;
                }
                selectCharacter = selectCharacter >> 1;
                for (int j = 0; j < 6; j++) {
                        word[j] = wordList[wordIndex][j];
                        if (selectCharacter & (char)1 == 1) {
                                int noiseBit = rand() % 6;
                                char bitTemp = 1;
                                for (int k = 0; k < noiseBit; k++) {
                                        bitTemp = bitTemp << 1;
                                }
                                if ((bitTemp & word[j]) == bitTemp) {
                                        word[j] = word[j] & ~bitTemp;
                                }
                                else {
                                        word[j] = word[j] | bitTemp;
                                }
                        }
                        selectCharacter = selectCharacter >> 1;
                }
                char out[6];
                find(word, out);
                int result = check(word, out);
                if (result == 0) {
                        score = 0;
                }
        }
        return score;
}
int main() {
        int score = 0;
        for (int tc = 1; tc <= TC_COUNT; tc++) {
                createAnimalCharacter();
                init(characterList, characterCnt);
```

```
                createAnimalWord();
                word(wordList, wordCnt);
                score += findCall();
        }
        printf("score : %d₩n", score);
        return 0;
}
```