

프로그래밍 과제 03

- 아래의 프로그램은 먼저 m 개의 사전식 순서로 정렬된 영문자열과, 다시 n 개의 사전식 순서로 정렬된 문자열을 입력받아 각각을 벡터 `first`와 `second`에 저장한다. 함수 `sorted_merge`는 두 벡터에 저장된 문자열을 벡터 `first`에 정렬된 상태로 합병하는 일을 한다. 이 함수를 완성하라. 아래 프로그램에서 `sorted_merge`함수의 내부를 완성하는 것 이외의 부분을 변경해서는 안된다. 라이브러리가 제공하는 `sort`함수를 사용해서는 안된다.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

void sorted_merge(vector<string> &first, vector<string> &second) {
    // Code here
}

int main() {
    int m, n;
    string str;
    vector<string> first, second;
    cin >> m;
    // m개의 영문자열이 사전식 순서로 정렬되어 입력된다.
    for (int i=0; i<m; i++) {
        cin >> str;
        first.push_back(str);
    }
    cin >> n;
    // n개의 영문자열이 사전식 순서로 정렬되어 입력된다.
    for (int i=0; i<n; i++) {
        cin >> str;
        second.push_back(str);
    }

    sorted_merge(first, second);

    // m+n개의 문자열이 사전식 순서로 정렬되어 출력되어야 한다.
    for (auto item: first)
        cout << item << " ";
    cout << endl;
    return 0;
}
```

입력 예	출력
5 10 13 17 19 22 4 11 15 17 23	10 11 13 15 17 17 19 22 23
8 and artist boys closest dirty essential global rotation 6 artistic broadcast close dry sense yield	and artist artistic boys broadcast close closest dirty dry essential global rotation sense yield
4 bcd cccd free glue 2 add cast	add bcd cast cccd free glue

2. 매개변수로 두 정렬된 정수 벡터 A 와 B 를 받아서 A 가 $(A \cup B) - (A \cap B)$ 가 되도록 하는 함수를 작성하라. 단, A 와 B 각각에는 중복된 정수가 없다고 가정한다. 라이브러리가 제공하는 **sort** 함수를 사용해서는 안된다. 아래 프로그램에서 `compute_set` 함수의 내부를 완성하는 것 이외의 부분을 변경해서는 안된다.

```
#include <iostream>
#include <vector>
using namespace std;

void compute_set(vector<int> &A, vector<int> &B) {
    // Code here
}

int main() {
    int m, n, k;
    vector<int> first, second;
    cin >> m;
    // m개의 정수가 오름차순으로 정렬되어 입력된다.
    for (int i=0; i<m; i++) {
        cin >> k;
        first.push_back(k);
    }
    cin >> n;
    // n개의 정수가 오름차순으로 정렬되어 입력된다.
    for (int i=0; i<n; i++) {
        cin >> k;
        second.push_back(k);
    }

    compute_set(first, second);

    for (auto item: first)
        cout << item << " ";
    cout << endl;
    return 0;
}
```

입력 예	출력
6 3 7 11 13 16 21 4 1 8 11 12 12 1 3 6 9 11 14 17 23 31 37 39 41 8 1 3 12 17 24 37 39 44	1 3 7 8 12 13 16 21
6 1 2 3 4 5 8 4 1 2 4 9	6 9 11 12 14 23 24 31 41 44
	3 5 8 9

3. 강의 1.7절에서 다룬 `IndexMaker` 프로그램을 다음과 같이 수정하라.
- 단어의 앞뒤에 붙은 소수점, 쉼표 등의 특수기호와 숫자는 모두 제거한다.
 - 모든 단어를 소문자로 변환한다.
 - 길이가 3미만인 단어들은 제외한다.
 - 인덱스에 단어들이 사전식 순서로 정렬한다.
 - 단어를 검색하면 그 단어가 등장한 라인번호들이 아니라 실제 라인들을 라인번호와 함께 한 줄에 하나씩 출력한다. 이를 위해서 파일의 라인들을 프로그램 내에 하나의 벡터로 저장하고 있어야 할 것이다. 하나의 단

어가 한 라인에 여러 번 등장하는 경우에도 동일 라인이 여러 번 출력되어서는 안된다. 검색에서 대소문자 구분은 하지 않는다.

f. “saveas” 명령은 구현하지 않는다.

아래의 예는 텍스트 파일 text.txt에 대한 실행 예이다.

FIND 명령의 예	출력
\$ find income	<p>The word income appears 10 times in lines: 1: on jobs and income at the inaugural AI Safety Summit in the U.K. in November. 5: a system of “universal high income.” 20: Musk’s concept of “universal high income” appears to be an evolution of the universal 21: basic income (UBI) idea supported by other tech leaders like Sam Altman. 23: “We won’t have universal basic income. We’ll have universal high income,” 26: first time Musk addressed the topic. In 2018, he posted on X: “Universal income 30: universal basic income, especially in light of the success of the expanded child 39: No country has introduced a universal basic income sufficient for essential needs, 43: of “universal high income” could ensure that no one falls below a certain income floor, 46: Musk’s positive outlook contrasts with recent critiques of universal basic income,</p>
\$ find universal	<p>The word universal appears 10 times in lines: 5: a system of “universal high income.” 20: Musk’s concept of “universal high income” appears to be an evolution of the universal 23: “We won’t have universal basic income. We’ll have universal high income,” 26: first time Musk addressed the topic. In 2018, he posted on X: “Universal income 30: universal basic income, especially in light of the success of the expanded child 39: No country has introduced a universal basic income sufficient for essential needs, 43: of “universal high income” could ensure that no one falls below a certain income floor, 46: Musk’s positive outlook contrasts with recent critiques of universal basic income, 48: credit and the Alaska dividend. Whether Musk’s AI-driven “universal high-income” 51: But his comments, alongside the evidence from recent universal basic income-related policies,</p>

FIND 명령의 예	출력
\$ find Musk	The word musk appears 6 times in lines: 0: Elon Musk made some striking predictions about the impact of artificial intelligence (AI) 8: where no job is needed,” Musk told U.K. Prime Minister Rishi Sunak. 13: Musk seemed optimistic about what he termed a “protopian” AI-driven future. 24: Musk said, though he did not explicitly define the difference. “In some sense, 26: first time Musk addressed the topic. In 2018, he posted on X: “Universal income 53: what that moment is,” Musk said, indicating that the world may need to prepare for

4. 입력 파일 `board.txt`에 오목판의 상태가 주어진다. 파일의 첫 줄에는 바둑판의 크기 $N \leq 19$ 가 주어지고, 이어진 N 줄에는 각 줄마다 N 개의 정수 0, 1, 혹은 2가 주어진다. 0은 빈자리를 표시하고, 1은 검은 돌, 2는 흰 돌을 표시한다. 주어진 상태가 검은 돌이 이긴 상태인지, 흰 돌이 이긴 상태인지, 혹은 아직 아무도 못 이긴 상태인지 검사하여 Black, White, 혹은 Not Finished라고 출력하는 프로그램을 작성하라. 둘 다 이긴 상태는 없다고 가정한다. 참고로 오목 게임은 네 돌이 수평, 수직, 혹은 대각선 방향으로 연속해서 5개가 놓이면 이기는 게임이다.

입력 예	출력
16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 2 0 2 0 0 0 0 0 0 0 0 0 0 2 0 1 1 1 2 2 0 2 0 0 0 0 0 0 0 1 1 0 1 0 1 0 2 0 0 0 0 0 0 0 0 2 2 1 1 1 0 2 0 1 0 0 0 0 0 0 0 0 1 0 1 0 2 0 0 0 0 0 0 0 0 0 0 2 0 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	White
10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 2 0 1 0 0 1 0 0 0 1 2 2 2 2 1 0 0 0 1 0 2 1 1 1 2 0 0 0 0 2 1 0 1 2 1 0 1 0 0 0 2 2 1 1 0 1 0 0 0 0 0 2 2 0 1 0 1 0 0 1 2 2 2 2 1 2 0 2 0 0 0 0 0 1 0 0 0 0	Not Finished

입력 예	출력
10	[3, 5]
5 3 15 3	[3, 12]
1 5 7 5	[6, 3]
7 7 16 7	[6, 5]
8 10 15 10	[9, 7]
3 12 12 12	[9, 10]
3 2 3 14	[9, 12]
6 1 6 9	[11, 7]
9 6 9 13	[11, 10]
11 5 11 11	[14, 3]
14 1 14 8	[14, 7]

6. [Self avoiding walk] 2차원 평면에서 원점 (0,0)에서 출발한다. 사용자가 현재의 위치에서 상하좌우 어떤 한 방향으로 얼마 만큼 이동하라는 명령을 내리면 그렇게 이동한다. 명령은 두 음이 아닌 정수로 표현된다. 우선 방향은 0, 1, 2, 3으로 표시하고 0은 y좌표가 증가하는 방향, 1은 x좌표가 증가하는 방향, 2는 y좌표가 감소하는 방향, 그리고 3은 x좌표가 감소하는 방향이다. 예를 들어 2 7은 y좌표가 7만큼 감소하는 위치로 이동하라는 명령이다. 프로그램은 사용자가 현재까지 이동한 궤적을 기억하고 있어야 한다. 사용자가 내린 명령대로 이동했을 때 만약 지금까지 이동한 궤적과 교차하면 **invalid move**라고 출력하고 이동 명령을 거부한다. 만약 그렇지 않으면 명령대로 이동하고 이동한 점의 좌표를 출력한다. 사용자가 -1 -1을 입력할 때 까지 이 일을 계속한다. 사용자가 -1 -1을 입력하면 프로그램을 종료한다.

입력 예	출력
1 3	3 0
3 5	invalid move
2 7	3 -7
3 9	-6 -7
0 3	-6 -4
1 11	invalid move
1 8	2 -4
2 3	invalid move
0 5	invalid move
0 2	2 -2
-1 -1	

7. 우선 인터넷을 검색하여 오셀로 게임의 규칙을 이해한 후 게임을 구현하라. 사람과 컴퓨터가 대결하는 방식이며, 컴퓨터는 항상 상대방의 말을 가장 많이 잡을 수 있는 위치에 놓도록 만들어라. 사람이 놓을 수 없는 위치에 말을 놓으려 시도하면 적절한 메시지를 출력하고 다시 입력 받도록 만들어라. 보드는 8×8 크기의 2차원 배열 혹은 “벡터의 벡터”로 표현하고, 검은 돌은 1, 흰 돌은 2, 그리고 빈칸은 0으로 표시한다. 사람은 검은 돌, 컴퓨터는 흰 돌이며, 항상 사람이 먼저 두기 시작한다고 가정한다. 돌이 하나 놓일 때 마다 현재 보드의 상태를 화면에 출력하라. 컴퓨터의 플레이를 구현하기 위해서 임의의 상황에서 어떤 위치에 말을 놓았을 때 잡을 수 있는 상대 말의 개수를 카운트하는 함수를 작성하여 이용하라.

```
int countStoneToCapture(int x, int y, int color)
```

여기서 (x,y)는 돌을 놓으려는 위치이고, color는 놓으려는 돌의 색깔(흑 혹은 백)이다. (x,y) 위치에 색이 color인 돌을 놓았을 때 잡을 수 있는 상대편 돌의 개수를 계산하여 반환한다. 이 함수를 이용하면 어렵지 않게 컴퓨터의 플레이를 구현 할 수 있고, 또한 사람이 어떤 위치에 돌을 놓아도 되는지도 이 함수를 이용하여 판단할 수 있을 것이다. 필요하다면 이 함수의 매개변수나 반환값 등을 변경하여도 상관없다.

8. 사전파일(dictionary.txt)로 부터 단어들을 읽어 저장한다. 그리고 아래 그림과 같은 2차원 문자 그리드를 다른 파일(puzzle.txt)로부터 읽는다. 이 파일의 첫 줄에는 그리드의 크기 $N \leq 16$ 이 주어지고, 이어진 N줄에는 각 줄마다 N개의 영문 소문자가 한 칸씩 띄어져서 주어진다. 그리드의 행, 열 혹은 대각선의 총 8방향(순방향과 역방향 포함)으로 만들어질 수 있는 사전에 있는 모든 단어들을 찾아 출력하라.

puzzle.txt

a	e	c	a	b	h
v	b	d	f	r	l
k	u	e	o	u	f
s	e	s	o	w	d
l	m	s	t	n	s
h	e	o	u	t	e

dictionary.txt

```
abilities
ability
able
about
above
absence
absolute
absolutely
abuse
academic
accept
acceptable
accepted
accepting
accepts
...
```

입력 예(PUZZLE.TXT)	출력
6 a e c a b h v b d f r l k u e o u f s e s o w d l m s t n s h e o u t e	a be do foot feel us of or so on one me to too no he out use