프로그래밍 과제 03

1. 강의 1.7절에서 다룬 `IndexMaker` 프로그램을 다음과 같이 수정하라.
   a. 단어의 앞뒤에 붙은 소수점, 쉼표 등의 특수기호와 숫자는 모두 제거한다.
   b. 모든 단어를 소문자로 변환한다.
   c. 길이가 3미만인 단어들은 제외한다.
   d. 인덱스에 단어들이 사전식 순서로 정렬한다.
   e. 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 단어가 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.txt`에 대한 실행 예이다.

FIND 명령의 예	출력

`$ find income`	`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,`

FIND 명령의 예	출력

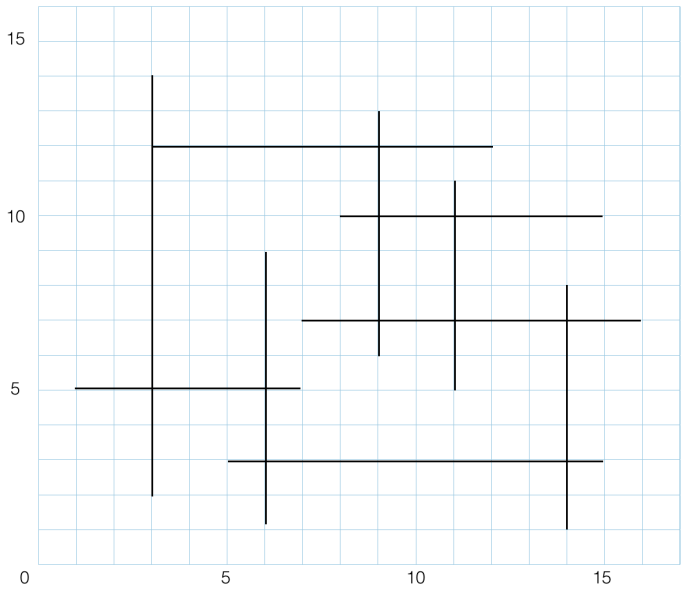
`$ find universal`	`The word universal appears 10 times in lines:`
	`5: a system of “universal high income.”`
	`20: Musk’s concept of “universal high income” appears to be an evolution of the universal`
	`23: “We won’t have universal basic income. We’ll have universal high income,”`
	`26: first time 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,`

`$ 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`

2. 입력 파일 `board.txt`에 오목판의 상태가 주어진다. 파일의 첫 줄에는 바둑판의 크기 $N \leq 19$가 주어지고, 이어진 N줄에는 각 줄마다 N개의 정수 `0`, `1`, 혹은 `2`가 주어진다. `0`은 빈자리를 표시하고, `1`은 검은 돌, `2`는 흰 돌을 표시한다. 주어진 상태가 검은 돌이 이긴 상태인지, 흰 돌이 이긴 상태인지, 혹은 아직 아무도 못 이긴 상태인지 검사하여 `Black`, `White`, 혹은 `Not Finished`라고 출력하는 프로그램을 작성하라. 둘 다 이긴 상태는 없다고 가정한다. 참고로 오목 게임은 내 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 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
15 0 0 0 0 2 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 2 0 1 0 1 0 2 0 0 0 0 0 1 0 2 0 1 0 1 2 1 0 2 0 0 0 0 0 2 0 1 2 1 1 1 2 2 0 2 0 0 0 2 0 2 0 1 0 1 0 2 0 1 0 0 0 1 0 0 0 2 0 1 2 1 0 2 0 0 0 0 0 0 0 0 0 0 2 0 2 1 2 0 0 0 0 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 1 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 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	Black

3. 입력으로 N개의 수직 혹은 수평 선분이 주어진다. 선분들간의 교차점의 좌표를 모두 계산하여 x좌표에 대한 오름차순으로 정렬하여 출력하는 프로그램을 작성하라. x좌표가 동일한 경우에는 y좌표가 작은 점을 먼저 출력한다. 입력의 첫 줄에는 선분의 개수 N이 주어지고, 이어진 N줄에는 각 줄마다 하나의 선분의 시작점과 끝점의 좌표가 주어진다. 수평 선분의 경우 x좌표가 작은 점이 먼저 주어지고, 수직 선분의 경우 y좌표가 작은 점이 항상 먼저 주어진다. 수직이나 수평이 아닌 선분이 주어지는 경우는 없다. 수평 선분끼리 만나거나 혹은 수직 선분끼리 만나는 경우는 교차점으로 간주하지 않는다. 이 문제를 해결하기 위해서 두 선분이 교차하는지 검사하는 함수 `intersect`를 만들어 사용하라.



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]

4. [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 move라고 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	

5. 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, `color`는 놓으려는 돌의 색깔(흑 혹은 백)이다. `(x,y)` 위치에 색이 `color`인 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.

6. 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

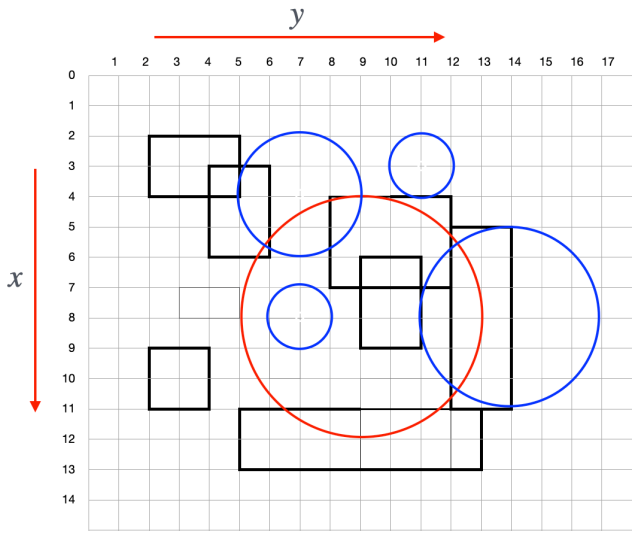
```
abilities
ability
able
about
above
absence
absolute
absolutely
abuse
academic
accept
acceptable
accepted
accepting
accepts
…
```

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	a be do foot feel us

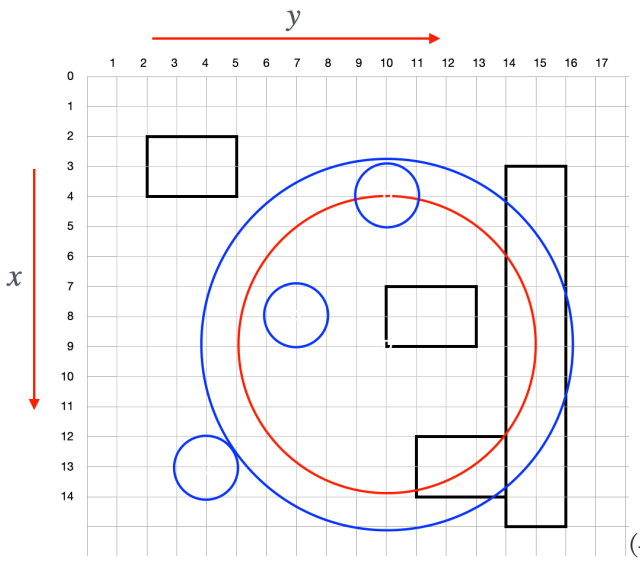
입력 예(PUZZLE.TXT)	출력
h e o u t e	of or so on one me to too no he out use

7. n개의 도형이 입력으로 주어진다. 도형의 종류에는 "좌표축에 평행한 직사각형"과 "원"이 있다. 이 도형들이 입력된 후 다시 추가로 하나의 원이 주어진다. 입력으로 주어진 도형들 중 추가로 주어진 원과 교차하는 도형들을 모두 찾아서 면적 순으로 정렬하여 출력하는 프로그램을 작성하라. 도형이 원의 내부에 포함되거나 혹은 반대로 원이 도형의 내부에 포함되는 경우도 교차하는 것으로 간주한다. 입력은 `input1.txt` 파일로 주어진다. 파일의 첫 줄에는 도형의 개수 $n \leq 1000$이 주어지고, 이어진 n줄에는 한 줄에 하나의 도형이 다음과 같은 형식으로 주어진다: 각 줄의 처음에는 먼저 도형의 종류를 나타내는 하나의 문자(사각형은 "`R`", 원은 "`C`")가 주어진다. 그런 다음 사각형의 경우 4꼭지점의 x좌표와 y좌표의 최소값과 최대값을 나타내는 4개의 정수가 x_{min}, x_{max}, y_{min}, y_{max}의 순서로 주어진다. 원의 경우 중점의 x좌표와 y좌표, 반지름을 나타내는 3개의 정수가 주어진다. 파일의 마지막 줄에는 추가로 주어지는 원의 중점의 x좌표, y좌표, 반지름을 나타내는 3개의 정수가 주어진다. 출력은 화면으로 한다. 면적 순으로 정렬된 도형들은 한 줄에 하나씩 입력과 동일한 형식으로 출력한다. 배열을 사용해서는 안되며, 원과 사각형을 표현하는 클래스 `Circle`과 `Rect`를 정의하여 사용하라. 두 클래스의 모든 데이터 멤버는 `private`으로 하라.

입력 예(INPUT1.TXT)	출력
11 R 2 4 2 5 R 3 6 4 6 C 4 7 2 R 9 11 2 4 R 4 7 8 12 C 8 7 1 R 6 9 9 11 C 3 11 1 C 8 14 3 R 5 11 12 14 R 11 13 5 13 8 9 4	C 8 7 1 R 3 6 4 6 R 6 9 9 11 R 4 7 8 12 R 5 11 12 14 C 4 7 2 R 11 13 5 13 C 8 14 3
8 R 2 4 2 5 C 4 10 1 C 8 7 1 R 7 9 10 13 C 13 4 1 R 12 14 11 14 C 9 10 6 R 3 15 14 16 9 10 5	C 4 10 1 C 8 7 1 R 7 9 10 13 R 12 14 11 14 R 3 15 14 16 C 9 10 6



(첫 번째 테스트 데이터, 빨간 원이 추가 입력된 원)



(두 번째 테스트 데이터)

8. N개의 좌표축에 평행한 직사각형들이 입력으로 주어진다. 모든 사각형들을 포함하는 가장 작은 원을 찾아서 중점의 좌표와 반지름을 출력하는 프로그램을 작성하라. 입력은 `input2.txt` 파일로 주어지고, 파일의 첫 줄에는 사각형의 개수 N이 주어지고, 이어진 N줄에는 한 줄에 하나의 사각형이 주어진다. 각각의 사각형은 4꼭지점의 x좌표와 y좌표의 최소값과 최대값을 나타내는 4개의 정수가 x_{min}, x_{max}, y_{min}, y_{max}의 순서로 주어진다.

입력 예	출력
4 2 4 2 5 7 9 10 13 12 14 11 14 3 15 14 16	8.5 9 9.55249
6 3 6 4 6 9 11 2 4 4 7 8 12 6 9 9 11 5 11 12 14 11 13 5 13	8.29348 8.17391 6.7411

입력 예	출력
7 2 4 2 5 3 6 4 6 9 11 2 4 4 7 8 12 6 9 9 11 5 11 12 14 11 13 5 13	7.5 7.5 7.77817