CSED101. Programming & Problem solving

Spring, 2019

Programming Assignment #3

정우창(wcjeong@postech.ac.kr)

- **Due: 2019.05.03 23:59**
- ***Development Environment*: GNU C Compiler (GCC) and Vi Editor (Editor is optional)**

- ***제출물***
  - **C Code file** (assn3.c)
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
  - **보고서 파일** (.doc(x) or .hwp) 예) assn3.doc(x) 또는 assn3.hwp
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - **리눅스 서버에 접속하는 것부터 시작해서 프로그램 컴파일 및 실행하는 과정까지를 화면 캡쳐하여 보고서에 포함시키고 간단히 설명 할 것!!**
    - **명예서약(Honor code)**: 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
    - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- ***주의사항***
  - 각 문제에 해당하는 요구사항을 반드시 지킬 것.
  - 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
  - 각 문제에 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다.
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
  - 과제 작성시 전역변수는 사용할 수 없으며, 사용자 정의 함수를 적절히 작성하도록 한다.
  - 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem: Linux Ant Game

(목적)

- 2차원 배열의 선언과 사용을 익힌다.
- 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힌다.
- 텍스트 파일 입출력을 익힌다.

(주의사항)

- 파일 이름은 **"assn3.c"**로 저장 할 것
- 보고서는 **"assn3.doc"** or **"assn3.hwp"**로 저장 할 것
- 전역변수, goto문, 구조체는 사용하지 않는다.
- 프로그램 구현 시, main() 함수를 호출하여 사용하지 않는다. 즉, 소스 코드 내에 main(); 이라고 호출하지 않는다.
- 적절히 사용자 정의 함수를 작성하여야 한다. (main() 함수 내에 모든 기능을 구현 시 감점)
- 과제에서 요구하는 사용자 정의 함수는 반드시 구현하여야 하며, 이외 필요한 함수는 적절히 정의해서 사용한다.
- 문제의 출력 형식은 채점을 위해 아래의 실행예시와 최대한 비슷하게 작성해 주세요.

(스토리)

linux user mode 세계에 살고 있는 개미들은 여느 때와 다를 것 없이 평화롭게 살아가고 있었다.

"개미는(뚠뚠) 오늘도(뚠뚠) 열심히 일을 하네(뚠뚠) .."

개미 노래를 부르면서 재미있게 놀고 있던 **linux user mode**의 개미들은 불현듯 **linux kernel mode**의 개미들에게 침공을 당한다. 어느덧 땅따먹기 전쟁으로까지 번지게 된 두 다른 mode 개미들의 싸움.. 과연 무사히 linux 세계에서의 전쟁을 끝내고 linux 세상의 평화를 되찾을 수 있을까?

(문제)

이번 과제에서는 동시턴제 땅따먹기 게임을 간소화하여 구현하려고 합니다.

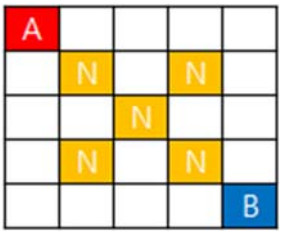

Figure 1 Linux Ant Game

빨간색으로 표시된 부분이 **Linux User Mode** 개미의 **Basement**이고 (**A**로 표시 - 사용자), 파란색으로 표시된 부분이 **Linux Kernel Mode** 개미의 **Basement**입니다 (**B**로 표시 - 컴퓨터). 그리고 노란색으로 표시된 부분이 A의 땅도, B의 땅도 아닌 **Neutral Region**입니다 (**N**으로 표시).

사용자가 A의 개미를, 컴퓨터가 B의 개미를 이동시키게 됩니다. 사용자 입력을 통해서 A에 위치한 개미들을 일련의 이동 규칙에 따라서 Region이나, Basement로 보낼 수 있습니다. 한쪽 편의 Basement가 Neutral 상태가 되거나, 상대방에게 점령당하게 되면, 게임이 끝나게 됩니다.

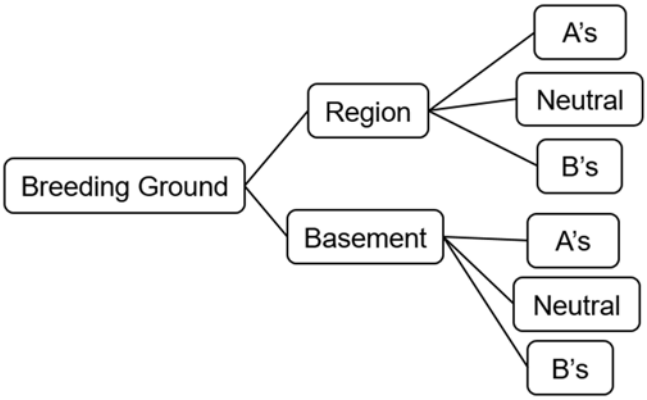

Figure 2 Breeding Ground의 정의

(게임 설명)

- 게임판은 최대 8 x 8의 크기를 가집니다. (단, 행의 수, 열의 수 ≥ 3)
- 사용자와 컴퓨터는 Breeding Ground로만 일정 이동 규칙에 따라서 개미를 보낼 수 있습니다. **Figure 1**에서 흰색으로 표시된 좌표에는 개미를 보낼 수 없습니다. Breeding Ground의 개수, 위치는 "map.txt"에서 결정됩니다.
- Breeding Ground는 크게 Region과 Basement로 나뉩니다.
- Basement는 A의 (사용자의) Basement와 B의 (컴퓨터의) Basement로 나뉘게 됩니다. A의 Basement는 맵의 가장 왼쪽 위에 위치하고, B의 Basement는 맵의 가장 오른쪽 아래에 위치하게 됩니다. 게임 시작 시에 A와 B 모두 Basement에 **10마리**의 개미를 가지고 시작하게 됩니다. (후에 설명이 나오게 되겠지만, Fight Turn을 거쳐갈 때, Basement에 위치한 나의 개미 수와 상대방의 개미 수가 같게 된다면, 그 Basement는 Neutral한 상태로 바뀌게 됩니다.)
- Region은 A의 Region, B의 Region, 그리고 Neutral Region으로 나뉘게 됩니다. "map.txt"를 통해 map을 읽어 들여서 가장 왼쪽 위, 가장 오른쪽 아래를 제외한 여러 지역에 Neutral Region을 생성합니다.
  (단, 1 ≤ map.txt를 읽어서 생성되는 Neutral Region의 개수 ≤ 7)
- 게임은 크게 **세 가지 Turn**으로 구성되어 있으며, Move turn ➔ Fight turn ➔ Breeding Turn 순서로 진행됩니다.
- Fight Turn에서 게임의 **승패 여부**가 결정됩니다.
  상대 Basement를 Neutral Basement로 바꾸거나, 나의 Basement로 만들게 되면 승리하게 됩니다. 반대로 나의 Basement가 Neutral Basement로 바뀌거나 상대의 Basement로 바뀌게 되면 패배하게 됩니다. 나와 상대방 모두 동시에 Basement를 빼앗기거나, Neutral Basement로 바뀌면 비기게 됩니다.
- 게임의 승패가 결정되면, 게임 결과 등의 정보를 파일(replay.txt)에 저장 후, 프로그램을 종료합니다.

- 아래 게임 설명과 실행 예제를 같이 보면 이해하기 쉽습니다.
- 본 과제 문서는 map 설명과 세 가지 Turn에 대한 설명, 요구 사항, 예외 처리, 코드 작성, 실행 예제, 힌트 순으로 구성되어 있습니다.

(1) map 설명과 Move Turn 설명

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)

Figure 3 map의 좌표 표시

- **Figure 3**에서 표시되어 있듯이 map의 세로 방향 Index를 Row, 가로 방향 Index를 Column이라고 하겠습니다. (row, column)
- A의 Basement 좌표 : (0, 0), B의 Basement 좌표: (맵 행의 수 - 1, 맵 열의 수 - 1) (위의 그림 3에서는 (4, 4))
- Move Turn에 사용자와 컴퓨터는 두 가지 **이동 규칙**에 따라 개미를 이동 시켜야 합니다. (개미 이동은 사용자와 컴퓨터가 동시에 선택합니다.)
- 자신이 소유하고 있는 Breeding Ground에서, 다른 Breeding Ground로 개미를 보낼 수 있습니다.
  단, 이동에 몇 가지 주의 사항이 있습니다. 주의 사항에 대해서 알려드리기 위해서, **Distance**라는 용어를 정의하도록 하겠습니다.

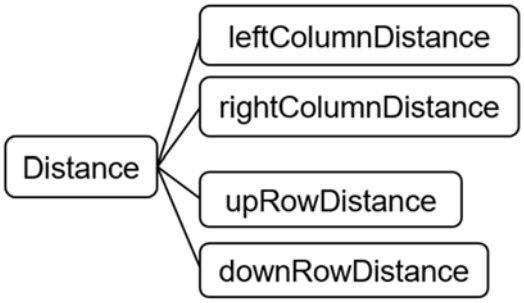

Figure 4 Distance의 정의

- leftColumnDistance : 같은 Row에서 왼쪽으로 가장 가까운 Breeding Ground와의 Column 차이
- rightColumnDistance : 같은 Row에서 오른쪽으로 가장 가까운 Breeding Ground와의 Column 차이
- upRowDistance : 위로 가장 가까운 Breeding Ground와의 Row 차이
- downRowDistance : 아래로 가장 가까운 Breeding Ground와의 Row 차이

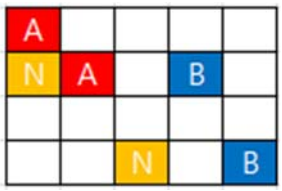

Figure 5 Distance 설명 예시 그림

Figure 5에서 (1, 1)의 A's region을 기준으로 설명하겠습니다.
(1, 1)의 A's region의

- leftColumnDistance : 1 ((1, 0)에 있는 중립 region이 제일 가깝다)
- rightColumnDistance : 2 ((1, 2)에 있는 B's region이 제일 가깝다)
- upRowDistance : 1 ((0, 0)에 있는 A's basement가 제일 가깝다)
- downRowDistance : 2 ((3, 2)에 있는 중립 region과 (3, 4)에 있는 B's basement가 제일 가깝다.)

이제 이동의 주의 사항입니다.
자신이 소유하고 있는 Breeding Ground에서,

1. 같은 Row 상에 있는 다른 Breeding Ground에 개미를 보내거나,
   A. 같은 Row에 네 개 이상의 Breeding Ground가 존재할 시에 leftColumnDistance만큼 떨어진 Breeding Ground나 rightColumnDistance만큼 떨어진 Breeding Ground에만 개미를 보낼 수 있습니다.
2. 다른 Row 상에 있는 다른 Breeding Ground에 개미를 보낼 수 있습니다.
   A. 다른 Row에 Breeding Ground가 존재할 시에 upRowDistance만큼 떨어진 Breeding Ground나 downRowDistance만큼 떨어진 Breeding Ground에만 개미를 보낼 수 있습니다.

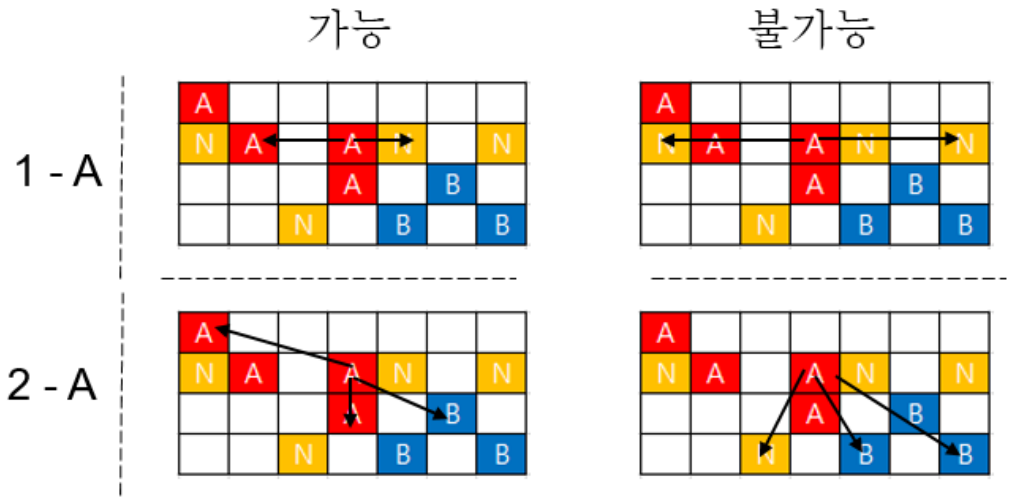

Figure 6 개미 이동의 주의 사항

1-A와 2-A 상황에 대한 장면도가 **Figure 6**에 도식화 되어있습니다. (1, 3)에서 이동 가능한 Breeding Ground를 생각했을 때 **Figure 6**에 나타나듯이 특정 Distance만큼 떨어진 Breeding Ground까지로만 이동 가능 합니다.

- 이동 규칙에 대해 알아보았으니 Move Turn이 어떤 순서로 진행되는지 알려드리겠습니다.

1. **사용자의 선택**

A. 출발지 선택

사용자가 소유하고 있는 Breeding Ground의 목록이 "Where ants start? "라는 문구와 함께 콘솔에 선택지 형태로 나타납니다. 사용자가 선택지 중 한 곳을 선택합니다.

B. 도착지 선택

사용자가 선택한 출발지에 해당하는 Breeding Ground로부터 이동 규칙에 따라 이동 가능한 Breeding Ground의 목록이 "Where ants go? "라는 문구와 함께 콘솔에 선택지 형태로 나타납니다. 사용자가 선택지 중 한 곳을 선택합니다.

C. 이동할 개미 수 선택

출발지에서 도착지로 몇 마리의 개미를 보낼 것인지 사용자로부터 입력을 받습니다. 먼저, "How many ants go?(0 ~ (출발지에서의 개미 마리수 - 1)) "라는 문구가 출력됩니다. (e.g. 출발지에서의 개미 마리수가 25마리일 때, "How many ants go?(0 ~ 24)" 라는 문구를 출력합니다.) 이후 사용자가 몇 마리의 개미를 보낼지 입력합니다.

2. **컴퓨터의 선택**

컴퓨터는 컴퓨터가 소유하고 있는 Breeding Ground 중 Random한 Breeding Ground에서 이동 규칙에 따라 이동 가능한 Random한 Breeding Ground까지 Random한 마리수의 개미(0 ~ 출발지에서의 개미 마리수 – 1)를 보낼지 선택합니다. 컴퓨터의 선택지는 콘솔에 보여지지 않습니다. 단, 선택지는 보이지 않아도 선택된 내용들은 출력됩니다.

3. **개미 이동**

출발지에서 보낼 개미의 수를 빼고, 도착지에서 받은 개미의 수를 더합니다. 이 때, A의 개미 1마리와 B의 개미 1마리가 같습니다. A의 개미 1마리와 B의 개미 1마리가 만나면 사라집니다. 아래 **Figure 7**의 왼쪽 그림에서 사용자가 (0, 0)에서 (1, 1)로 30마리, Computer가 (1, 1)에서 (0, 0)으로 20마리의 개미를 보냈다고 가정해봅시다. 그러면, **Figure 7**의 오른쪽 그림과 같이 나타나게 됩니다.

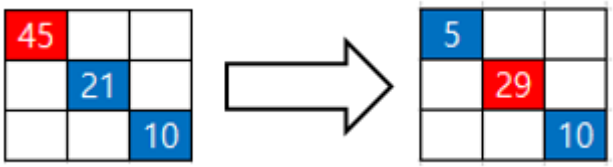

Figure 7 개미 이동 설명

(2) Fight Turn 설명

- Move Turn이 종료된 후의 Turn입니다.
- Fight turn에서 하는 일은

1. 현재 각 Breeding Ground에 얼마만큼의 개미가 남아있는지 확인하여,
2. 누구의 Breeding Ground인지를 결정하는 일입니다.
3. 또한 승패를 결정하는 Turn이기도 합니다.

- A의 개미가 한 마리라도 존재하면 A의 Breeding Ground, B의 개미가 한 마리라도 존재하면 B의 Breeding Ground, 개미가 존재하지 않으면 Neutral Breeding Ground입니다.

- 게임의 승패 결정 조건은 아래와 같습니다.
  1. 승리 : B의 Basement가 A의 Breeding Ground나 Neutral Basement로 바뀌면 승리(Win)
  2. 패배 : A의 Basement가 B의 Breeding Ground나 Neutral Basement로 바뀌면 패배(Lose)
  3. 무승부 : B의 Basement가 A의 Breeding Ground나 Neutral Basement로 바뀜과 동시에 A의 Basement가 B의 Breeding Ground나 Neutral Basement로 바뀌면 무승부(Draw)

- Region의 개수나 현재 가지고 있는 개미의 마리 수는 게임의 승패에 아무런 관련이 없습니다. Basement의 점거 여부가 게임의 승패를 결정합니다. Fight Turn에서 승패가 결정되지 않으면, Breeding Turn으로 넘어갑니다.

(3) Breeding Turn 설명

- Fight Turn이 종료된 후의 Turn으로, 문자 그대로 번식하는 Turn입니다.
- 두 가지 번식 규칙에 따라서 Breeding Ground의 개미의 마리수가 증가 됩니다. (Neutral region 제외)
  1. Basement : Basement의 개미 마리수가 2 증가합니다.
  2. Region : Region의 개미 마리수가 1 증가합니다.
  3. Neutral region의 개미 마리수는 증가하지 않습니다.
- Breeding Turn이 종료된 이후에는 다시 Move Turn으로 넘어갑니다.

(4) 요구 사항

- 게임에 대한 설명은 Turn 설명에서 어느 정도 드렸습니다. 요구사항에 대해 간단히 적어보도록 하겠습니다.
  1. 예외 처리(Exception Handling) (아래에서 자세히 알려드리겠습니다.)
  2. 코드 작성시 과제 문서에서 제시한 함수를 사용해 주십시오.(코드 작성 부분 참조)
  3. 출력 형식, 파일명, 글씨 색깔 등을 지켜주십시오. (실행 예제 참조)
- 이외에도 게임 설명에 맞는 게임 구현은 필수입니다.

(5) 예외 처리

- 게임을 개발하고 실행하다 보면, 게임에서 의도하지 않은 파일/사용자 입력이 들어올 때가 존재합니다. 이런 예외 상황들 때문에 에러나 버그가 발생할 수 있습니다. 비정상적 동작을 막기 위해서 예외 처리를 해야만 합니다. assert()함수를 사용해서 예외 처리할 수도 있지만 이번 과제에서는 if, else문을 활용하여 예외 처리 해주십시오.

- 아래와 같은 상황이 발생했을 때, 적절한 에러메시지를 콘솔에 출력 한 이후에 상황에 따라 사용자 입력을 다시 받거나 프로그램을 종료하게 됩니다. 에러메시지에는 무엇을 적더라도 상관없습니다. 그러나 영어로 적어주시고, 한 번에 보았을 때 무슨 연유로 프로그램이 종료되었는지 알기 쉽게 적어주십시오. 아래의 명시된 예외 상황이 아닌 다른 예외 상황 같은 경우 채점 대상이 아닙니다.

- **"map.txt" 파일에 대한 예외처리**

  "map.txt" 파일이 존재하지 않을 때, 적절한 에러 메시지를 콘솔에 출력한 이후에 프로그램을 종료합니다. "map.txt" 파일 구성(10쪽 참고)이 맞지 않은 파일에 대해서는 고려하지 않습니다.

  (예외 처리 화면 캡쳐)

  "map.txt" 파일이 존재하지 않을 때, 적절한 에러 메시지를 출력하고 프로그램이 종료되는지 화면 캡쳐하여 보고서에 넣어주시기 바랍니다.

- **Move Turn에서의 예외처리**

  Move Turn에서는 출발지, 도착지, 개미의 수를 순서대로 입력 받아 게임을 진행합니다. 이 때, 사용자로부터 적절하지 못한 입력 값에 대해서, 아래의 설명과 같은 적절한 메시지 출력 후 다시 입력 받아 게임을 계속 진행하도록 합니다.

  1. 출발지 선택 시 예외 처리

     선택지에 없는 범위의 정수나 실수, 문자가 입력되면 "Please type an integer from 1 to (내가 소유하고 있는 Breeding Ground의 수) : " 라는 메시지와 함께 사용자로부터 선택지 번호를 다시 입력 받습니다.

     예) 내가 소유하고 있는 Breeding Ground의 개수가 3이라면, "Please type an integer from 1 to 3 : " 라는 메시지를 화면에 출력 후, 선택지 번호를 입력 받습니다.

     (예외 처리 화면 캡쳐)

     선택지에 없는 범위에 정수가 입력 되었을 때, 실수가 입력되었을 때, 문자가 입력되었을 때 각 경우에 대해서 화면 캡쳐하여 보고서에 넣어주시기 바랍니다. Move Turn 의 경우, 이 항목에 대한 예외처리만 화면 캡쳐하여 보고서에 넣으면 됩니다.

  2. 도착지 선택 시 예외처리

     선택지에 없는 범위의 정수나 실수, 문자가 입력되면 "Please type an integer from 1 to (출발지로부터 이동 규칙에 따라 이동 가능한 Breeding Ground의 수) : " 라는 메시지와 함께 사용자로부터 선택지 번호를 다시 입력 받습니다.

     예) 출발지로부터 이동 규칙에 따라 이동 가능한 Breeding Ground의 수가 1이라면, "Please type an integer from 1 to 1 : " 라는 메시지를 화면에 출력 후, 선택지 번호를 입력 받습니다.

  3. 개미 수 선택 시 예외처리

     출발지에 존재하는 개미의 수를 벗어난 범위의 정수나, 실수, 문자가 입력되면 "Please type an integer from 0 to (출발지에 존재하는 개미 마리수 - 1) : " 라는 메시지와 함께 사용자로부터 다시 선택지 번호를 입력 받습니다.

     예) 출발지에 존재하는 개미의 숫자가 30이라면, "Please type an integer from 0 to 29 : " 라는 메시지가 콘솔에 출력되어야 합니다. 그리고 선택지 번호를 입력 받아야 합니다.

(6) 코드 작성

반드시 작성해야 하는 함수는 다음과 같습니다. 이 함수들 말고도 자유롭게 추가로 함수를 정의해서 사용하도록 하세요. 아래 명시된 매크로 이름, 함수 이름, 변수 이름은 바꾸지 말아주세요. 함수 이름과 변수 이름의 대소문자나 철자에 유의해주시기 바라겠습니다. 적절한 주석은 필수입니다. (필요한 매개변수가 있으면 추가하여 구현해주세요.)

■ int main()

아래의 2차원 배열은 main 함수에서 선언 후, 사용해야 합니다.

- char map[MAXROW][MAXCOLUMN]; // 게임 판을 나타내는 배열
- int antCounts[MAXROW][MAXCOLUMN]; // 각 Breeding Ground의 개미 수를 저장하는 배열

MAXROW와 MAXCOLUMN은 적절한 상수로 define하여 사용하도록 합니다.
아래의 사용자 정의 함수들의 경우, main 함수에서 호출해서 사용합니다.

■ 파일 입력 관련

- **int ReadMap(char map[][MAXCOLUMN], int antCounts[][MAXCOLUMN], int * row, int * column, int * regionCount)**
  "map.txt" 파일을 읽는 함수입니다. 인자로 받은 map 배열과 antCounts 배열에 **Figure 8** 형태로 출력될 수 있도록 각각 게임판과 개미 수를 저장합니다. row와 column에는 각각 row의 개수와 column의 개수를 call by reference로 저장하고, regionCount에는 파일로부터 읽어 들인 Neutral Region의 개수를 저장해주십시오. 만일 "map.txt" 읽기를 실패한 경우 0을 반환하고, 그렇지 않다면 1을 반환합니다.

■ 콘솔 출력 관련

- **void PrintMap(char map[][MAXCOLUMN], int row, int column)**
  출력 형식에 맞게 map을 출력하는 함수입니다.

- **void PrintAntCounts(char map[][MAXCOLUMN], int antCounts[][MAXCOLUMN], int row, int column)**
  출력 형식에 맞게 map의 개미 숫자를 출력하는 함수입니다.

■ Turn 관련

Turn 관련해서 적절한 사용자 정의 함수를 세 개 이상 정의하여 구현해주십시오. Turn 관련 사용자 함수는 주석을 통해서 명시해주시고, 정의한 함수가 어떤 기능을 이행하는지 적어주십시오. 각 함수는 main 함수에서 정의된 map 배열과 antCounts 배열을 매개변수로 포함하고 있어야 합니다.

■ 파일 출력 관련

- **void PrintResult(char map[][MAXCOLUMN], int antCounts[][MAXCOLUMN], int row, int column, int gameResult)**
  출력 형식에 맞춰서 "replay.txt"를 출력하는 함수입니다.

(7) 실행 예제

- "assn3.c"로부터 컴파일 된 프로그램을 실행하면 "map.txt"에서 정보를 읽어 아래 **Figure 8**과 같이 Breeding Ground의 점거 상황과 개미 수를 보여주는 게임 판을 출력합니다.

map.txt 예시
4 5
3
1 1
1 3
3 3

파일의 첫 줄에는 Row의 크기와 Column의 크기가 순서대로 주어집니다. 그 다음 줄에는 생성되는 Neutral Region의 개수가 주어지고, 그 다음 줄부터는 Neutral Region의 좌표들이 순서대로 주어집니다.

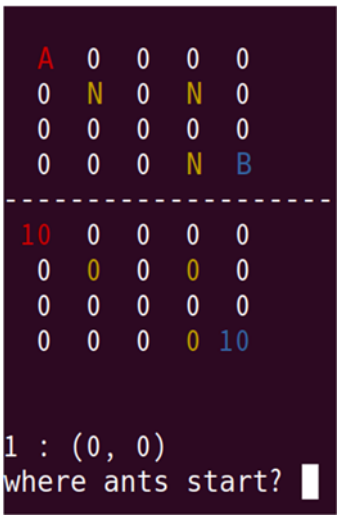

Figure 8 게임 시작 화면

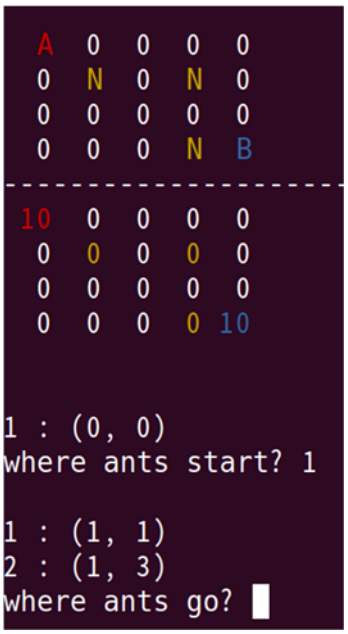

Figure 9 도착지 입력

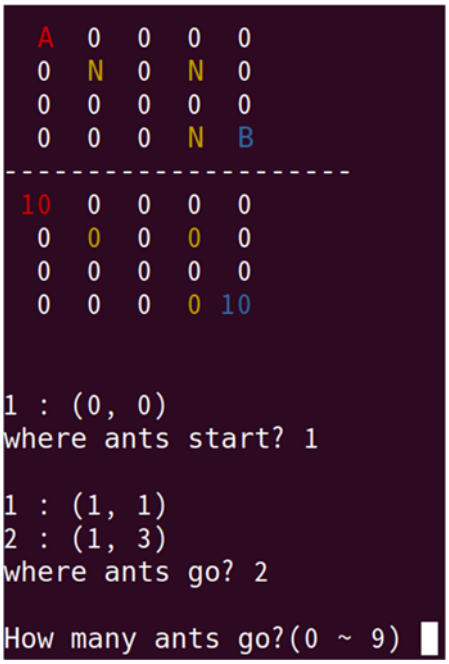

Figure 10 개미 수 입력

- **Figure 8**에서, 사용자가 출발지로 삼을 수 있는 곳의 보기가 모두 출력됩니다. 그리고 사용자로부터 출발지의 보기 번호를 입력 받습니다. 보기 번호는 1부터 시작해서 보기를 제시 바랍니다. (앞에서 언급한 예외 처리 항목에 대한 처리가 필요한 부분입니다.)
- 사용자로부터 출발지를 입력 받게 되면, **Figure 9**와 같이 선택한 출발지로부터 이동 규칙에 따라 이동할 수 있는 도착지의 목록이 자동으로 계산되어 출력됩니다. 마찬가지로 보기 번호를 사용자로부터 입력 받습니다. 가능한 출발지나 도착지가 여러 곳일 경우, 모두 출력하되 그 순서는 상관 없습니다.
- 사용자로부터 도착지까지 입력 받았다면, 이후에는 얼마만큼의 개미들을 출발지에서부터 도착지까지 보낼지 정하게 됩니다.

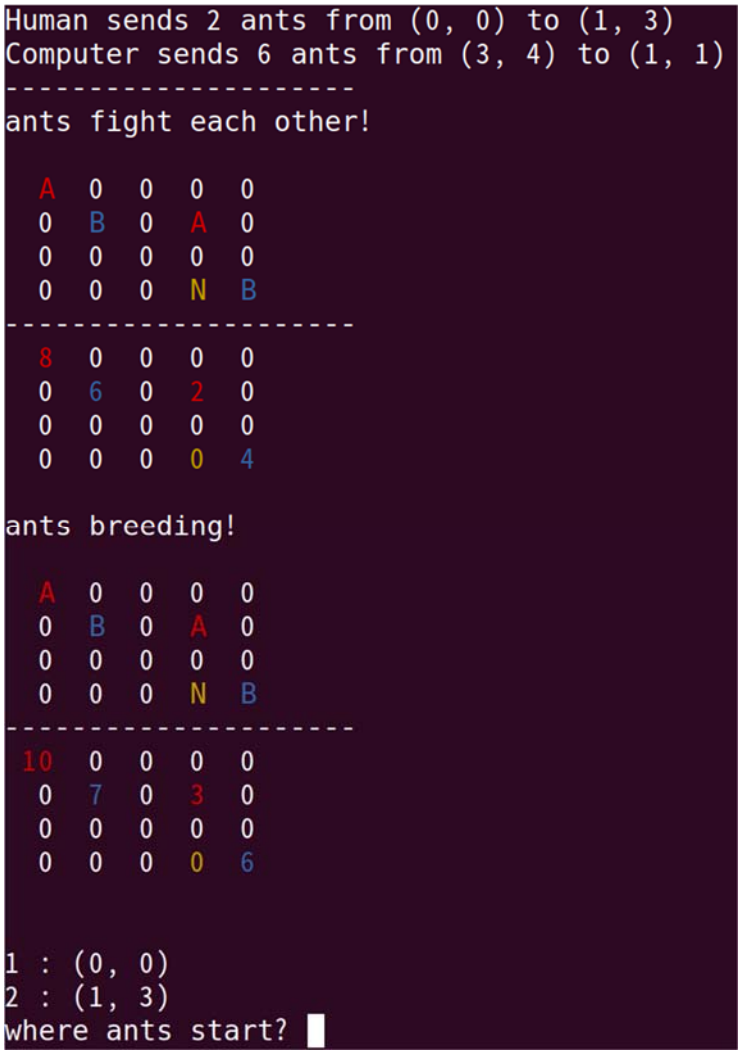

Figure 11 사용자 입력 이후

- **Figure 10** 의 상황에서 이동할 개미의 수로 숫자 2를 입력하였습니다. 화면이 지워지고 Figure 11과 같은 화면이 출력 됩니다. 화면의 가장 윗부분에 사람(사용자)과 컴퓨터가 몇 마리의 개미를 어디에서 어디로 보냈다는 문구가 출력됩니다.
- 컴퓨터는 random하게 출발지, 도착지, 개미 수를 선택하게 되는데, 컴퓨터의 선택지는 콘솔 상으로 보여지지 않습니다. 단, 선택지는 보이지 않아도 선택된 내용들은 출력됩니다. (Ex. 위와 같이 Computer sends 6 ants from (3, 4) to (1, 1))
- Fight Turn과 Breeding Turn에서의 상황이 콘솔 상으로 보여지게 되고, 또 다시 사용자로부터 입력 받게 게임이 진행 됩니다.

- 게임 진행 중에 승패가 결정되면 게임이 종료됩니다. 게임이 종료되기 이전에 system("clear")를 이용해서 화면 지우기를 한 이후에 **Figure 12, 13, 14**와 같이 게임 결과를 화면에 출력합니다.

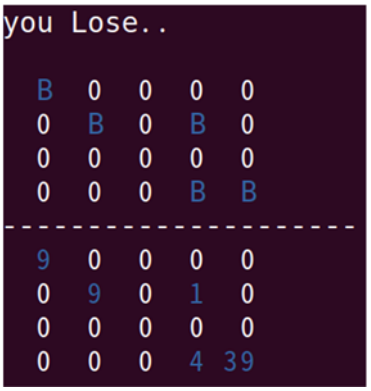

Figure 12 게임 종료 상황 1

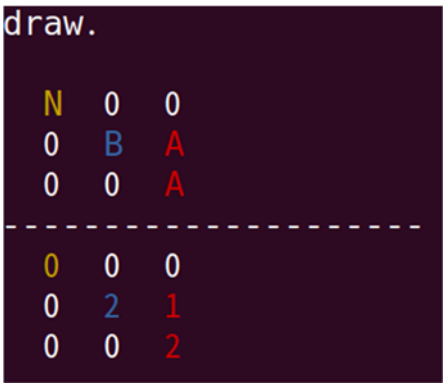

Figure 13 게임 종료 상황 2

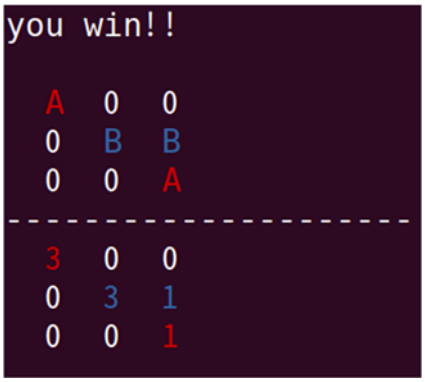

Figure 14 게임 종료 상황 3

- 또한, 게임의 결과 등의 정보를 파일(replay.txt)에 저장 후, 프로그램을 종료합니다. **Figure 12, 13, 14**와 같이 게임이 종료되었을 때, 파일 출력을 통해 생성되는 replay.txt의 예시는 아래와 같습니다. replay.txt는 총 세 줄로 구성되어 있습니다.

replay.txt 예시 1	replay.txt 예시 2	replay.txt 예시 3
L 0 0 5 62	D 2 3 1 2	W 2 4 2 4

1. 첫번째 줄 : 승패 표시 – L(패배를 의미), D(무승부를 의미), W(승리를 의미)
2. 두번째 줄 : 사용자 최종 정보로 사용자가 소유한 Breeding Ground의 개수와 사용자가 소유하고 있는 총 개미의 마리수가 순서대로 파일에 출력됩니다.
3. 세번째 줄 : 컴퓨터 최종 정보로 컴퓨터가 소유한 Breeding Ground의 개수와 컴퓨터가 소유하고 있는 총 개미의 마리수가 순서대로 파일에 출력됩니다.

(8) 힌트

■ 리눅스에서 글자색 출력

ANSI ESC code를 사용하여 리눅스 터미널에 출력되는 글자의 색을 바꿀 수 있습니다. "<ESC>[<색상코드>m" 을 터미널에 출력하면, 이후에 출력되는 글자의 색이 색상코드에 따라 바뀌게 됩니다. ESC를 출력하기 위해서는 ASCII 코드 0x1b를 사용하면 된다. 색상코드는 다음과 같은 것들이 있습니다.

빨간색	31
노란색	33
파란색	34
RESET	0

한번 색상코드를 지정하면, 다른 색으로 다시 지정하거나, 초기화를 하기 전까지 해당 색이 출력됩니다. 아래의 코드는 빨간색으로 "HELLO WORLD"를 출력하는 예시입니다.

```c
#include <stdio.h>

int main()
{
    printf("\x1b"); // ESC 출력
    printf("[31m"); // 빨간색 색상코드 출력
    printf("HELLO WORLD\n"); // 원하는 문자 출력
    printf("\x1b\n"); // ESC 출력
    printf("[0m\n"); // 초기화 코드 출력


    return 0;
}
```

다음과 같이 매크로를 사용하면, 좀 더 편하게 색을 지정할 수 있습니다.

```c
#include <stdio.h>
#define RED "\x1b[31m"
#define RESET "\x1b[0m"

int main()
{
    printf(RED); // 빨강 색상코드 지정
    printf("HELLO WORLD\n"); // 원하는 문자 출력
    printf(RESET); // 색상 초기화

    printf(RED "HELLO WORLD\n" RESET); // 이렇게 한번에 붙여서 출력해도 됨
    return 0;
}
```

■ 화면 지우기

화면을 지우기 위해서는 stdlib.h 라이브러리 안의 system 함수를 사용해야 합니다. 다만, 윈도우의 비주얼 스튜디오가 리눅스의 프로그래밍 환경이 다릅니다. 그렇기에, system 함수로 넘겨줄 매개변수로 "cls"대신 "clear"를 사용합니다.

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("HELLO WORLD\n"); // 원하는 문자 출력
    system("clear"); // 윈도우의 경우 system("cls");
    return 0;
}
```

■ 배열 사용

- 배열의 크기를 선언할 때 변수를 사용하면 안 됩니다.
- 충분한 크기로 배열을 선언 한 뒤 필요한 부분만 화면에 출력해야 합니다.
- 2차원 배열을 함수의 매개변수로 넘겨주기 위해서는 arr[][3]과 같은 표현을 사용합니다.

```c
#define MAXROW 3
#define MAXCOLUMN 3

int ArrayTestFunc(int arr[][MAXCOLUMN], int testvalue);
int main()
{
    int arr[MAXROW][MAXCOLUMN] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
    ArrayTestFunc(arr, MAXROW);
    return 0;
}
```

■ 어싸인 팁

1. 이번 과제의 개발 환경은 리눅스입니다. 제출 전, 반드시 Linux 에서 제대로 컴파일 되고 동작되는지 확인하십시오. Linux 환경에서 제대로 실행되지 않으면 감점입니다.

2. 프로그램 작성 중 무한루프에 의하여 프로그램이 끝나지 않을 경우 Ctrl+C를 누르세요.

3. 코딩부터 시작하지 마십시오. 실행 할 프로그램에 대한 구조와 알고리즘을 먼저 설계한 후, 컴퓨터 앞에 앉아서 코딩을 시작하세요. 복잡하고 어렵게 코드를 작성하는 사람이 좋은 프로그래머가 아닙니다. 누가 보더라도 쉽게 프로그램의 흐름을 이해할 수 있는 가독성 좋은 쉬운 코드를 작성해 주기 바랍니다.