CSED101. Programming & Problem solving
Spring, 2020

Programming Assignment #3
(75 points)

김태훈(taehoon1018@postech.ac.kr)

■ *Due*: 2020.06.28 23:59

■ *Development Environment: Windows Visual Studio 2019*

■ *제출물*

- C Code files (assn3.c)
  - ➢ 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
- **보고서 파일** (assn3.docx, assn3.hwp 또는 assn3.pdf)
  - ➢ AssnReadMe.pdf 를 참조하여 작성할 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 채점을 위해 실행 예시와 최대한 비슷하게 작성해 주세요.
- 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.)
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem: 세균전

[목적]
이번 과제를 통하여 포인터와 동적 할당 사용법을 익힌다.

[문제]
세균전은 일종의 땅 따먹기 게임으로, 제한된 게임 판 위에서 더 많은 칸을 차지하는 플레이어가 승리하게 되는 프로그램을 작성한다.

[주의사항]

(1) 과제는 2차원 배열을 동적으로 할당하고 사용하기 위한 과제이므로, 게임 판은 반드시 2차원 배열을 동적으로 할당 후 사용 및 해제해야 한다. (마지막 페이지의 힌트를 참고할 것)

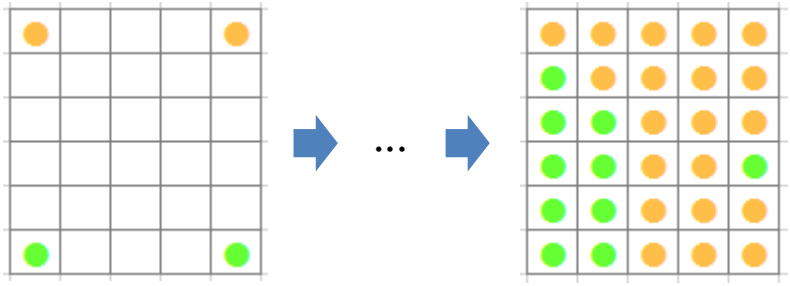
(2) 게임 진행 시, 엔터 없이 키입력을 하도록 한다. (마지막 페이지의 힌트를 참고할 것)

(3) 전역 변수, goto 문, string 관련 함수, 구조체는 사용할 수 없다.

(4) 모든 기능을 main 함수에서 구현한 경우 감점 처리한다. 기능적으로 독립됐거나 반복적으로 사용되는 기능은 사용자 함수를 정의해서 구현하도록 한다.

(5) 문제의 출력 형식은 채점을 위해 아래의 실행 예시와 일맥상통하도록 출력한다. (비슷하거나 형식이 이해된다면 감점 없음)

[게임 설명]
세균전은 2인용 게임으로 아래의 왼쪽 같은 게임 판으로 시작하여, 플레이어 2명이 번갈아 가면서 규칙대로 말을 이동 또는 복제하여 칸을 더 많이 차지하는 플레이어가 승리하게 된다.



(1) 보드의 크기는 N x M 행렬로 구성된다.
(2) 2명의 플레이어가 번갈아 가면서 자신의 순서에 말을 선택을 하고, 옮길 곳을 지정한다. 선택한 말에서의 거리가 한 칸 차이면 복제, 두 칸은 이동이 된다.
(3) 옮긴 곳(복제 또는 이동)을 중심으로 인접해 있는 상대방의 말은 모두 내 것이 된다.
(4) 두 플레이어 모두 옮길 곳이 없으면 게임이 종료된다.
(5) 게임이 종료 시, 더 많은 곳에 말을 둔 플레이어가 승리한다.

[실행 예시 및 설명]

(1) 초기 실행 화면
게임 판 생성을 위해 게임 판의 행과 열의 크기를 입력 받는다. 크기의 제한은 4부터 20까지이며, 범위 외의 입력에 대해서는 적절한 예외처리를 하고 다시 입력을 받도록 한다.

```
[ATAXX]
row: 5
col: 4
```

← 왼쪽 실행 예제의 빨간색 밑줄은 사용자 입력에 해당

그림 1. 초기 실행 화면

(2) 게임 판 초기화면

게임 판의 크기를 입력 받은 후, 화면을 모두 지우고 아래와 같이 게임 보드를 출력한다. 화면의 상단에는 각 플레이어들이 차지하고 있는 칸의 숫자가 출력된다.

- 초기 게임 판 설정
  - ① Player 1
    노란색 말(@)로 나타내고 (0, 0)과 (0, col-1)에 말 2개를 배치
    Player 1의 게임 시작 위치는 (0, 0)
  - ② Player 2
    연두색 말(@)로 나타내고 (row-1, 0)과 (row-1, col-1)에 말 2개를 배치
    Player 2의 게임 시작 위치는 (row-1, col-1)

※ 글자색은 9쪽의 “글자색 출력” 부분을 참고한다.

게임 시작 시, Player 1 이 항상 먼저 시작하도록 한다.

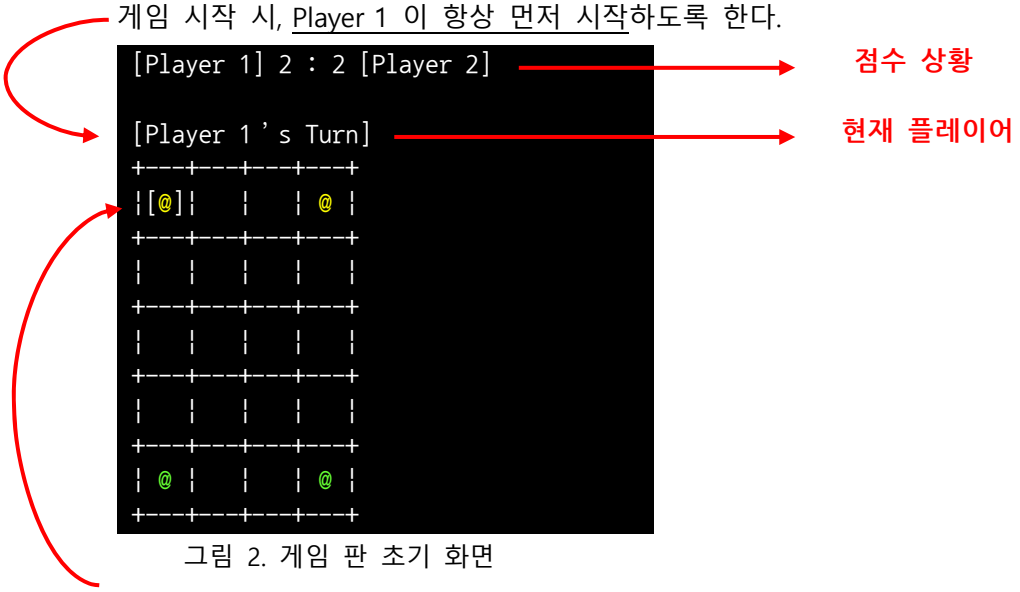


그림 2. 게임 판 초기 화면

그리고 Player 1의 게임 시작 위치가 [ ] 를 이용하여 표시되고 있다.

플레이어들의 게임 시작 위치는 자신의 순서에 마지막으로 자신이 말을 놓았던 장소를 기억하고 있다가 그 위치부터 커서([ ])가 나타나도록 한다.

(3) 플레이어들은 자신의 순서에서 방향키를 움직여서 게임 판에서 칸을 이동할 수 있다.

- 사용자 입력으로 사용되는 키는 다음과 같다.
  - i(위), j(왼쪽), k(아래), l(오른쪽): 커서를 움직이는 방향키

※ 단, 게임 판의 경계에서는 바깥으로 움직이려고 해도 커서가 움직이지 않는다. 커서가 그림 2처럼 (0, 0)위치에 있다고 하면 i 나 j를 눌러도 커서가 이동하지 않는다.

  - space 키:
    ① 자신의 말의 위치에서 스페이스를 누르면 옮길 말이 선택된다.
    선택된 말을 취소하고 다른 자신의 말을 선택하고 싶으면, 단순히 다른

말로 커서를 이동 후, 스페이스를 누르면 된다.

② 말이 선택된 경우, 이동(또는 복제)이 가능한 빈 칸에서 스페이스를 누르면 그 위치로 규칙에 따라 이동(또는 복제)이 된다.

※ 적절하지 않은 입력은 아무 일도 일어나지 않는다.

예)

- 위에서 명시한 키 외의 입력
- 다른 플레이어의 말의 위치에서 스페이스 입력
- 이동 불가능한 위치에서 스페이스 입력

그림 2에서 l(오른쪽)키를 누르면, 화면을 지운 후 그림3-1과 같이 출력된다. Player 1의 커서의 위치가 (0, 1)의 위치로 이동했음을 볼 수 있다.

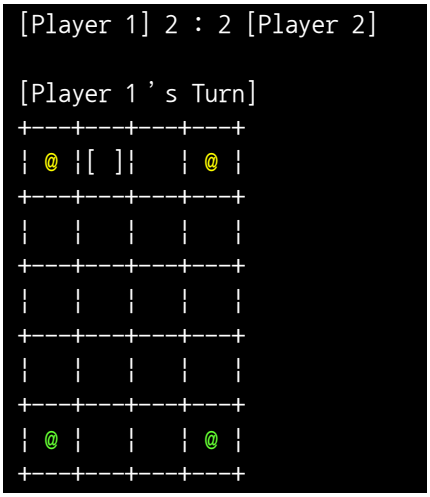


그림 3-1. 커서 이동

l(오른쪽)키 입력

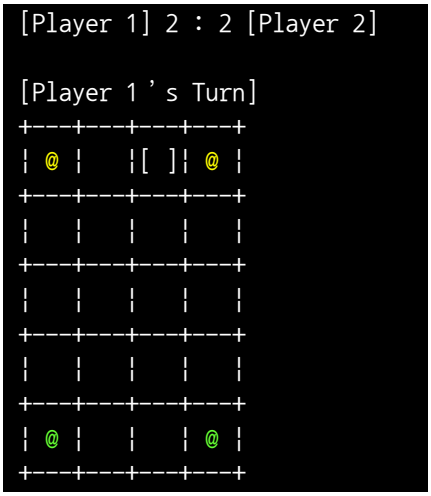


그림 3-2. 커서 이동

l(오른쪽)키 입력

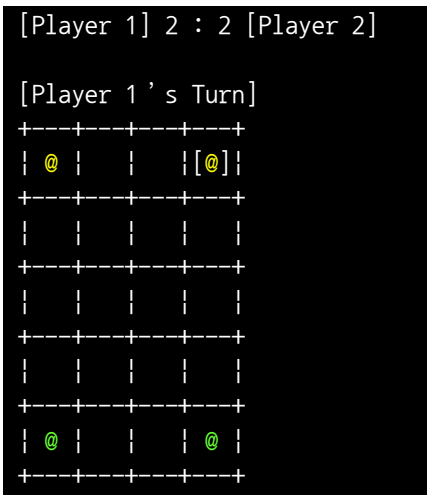


그림 3-3. 커서 이동

스페이스 키 입력

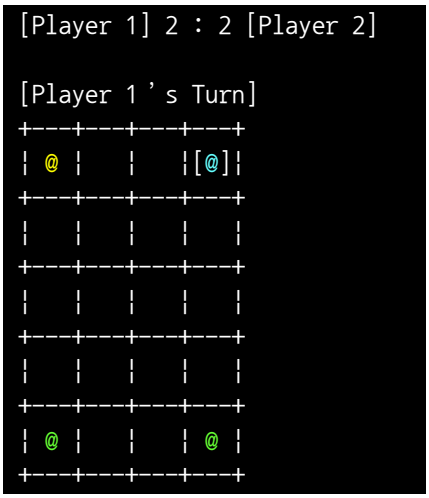


그림 3-4. 말 선택

그림 3. 이동 및 말 선택 예시

그림 3-1 에서 1키를 입력 후, 커서의 위치가 (0, 2)의 위치로 이동(그림 3-2), 또 다시 1키를 입력하여 (0, 3)의 위치로 커서가 이동(그림 3-3)된 것을 볼 수 있다.

(4) 말 선택 후 이동(또는 복제)

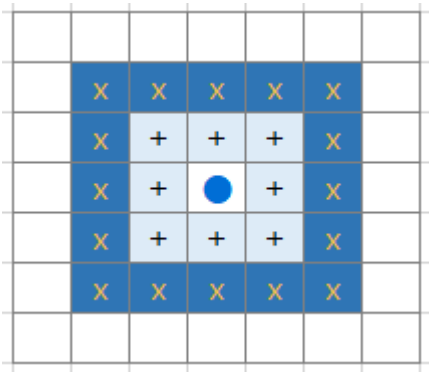
그림 3-3의 커서의 위치에는 Player 1의 말이 있으므로 '스페이스' 키를 입력하여 자신의 말을 선택할 수 있다(그림 3-4). 선택된 말은 하늘색(@)으로 표시한다.

선택된 말은 규칙에 따라 이동(또는 복제)를 할 수 있다.

- 말 이동 또는 복제 규칙

  말은 자신의 말(●)에서 한 칸 혹은 두 칸 거리에 있는 빈칸에 놓을 수 있다.

  - 가로, 세로, 대각선으로 한 칸 떨어진 곳(+ 표시)은 말이 복제되어 놓이고,
  - 두 칸 떨어진 곳(x 표시)은 그 지점으로 말이 이동한다.



말 선택 후, 복제 또는 이동하기 위해 다시 방향키(i, j, k, l)키를 사용하여 커서를 이동한다. 그림 3-4에서 말 선택 후, k(아래)키를 입력하면 그림 4-1과 같이 커서가 (1, 3)의 위치로 이동한다. 이 때 스페이스 키를 누르면, 선택된 말에서 한 칸 떨어진 곳이어서 그림 4-2와 같이 복제가 된다. 말이 선택된 상태에서 복제(또는 이동)가 가능한 빈칸에서의 스페이스 키 입력은 말을 그곳에 놓겠다는 의미이다.

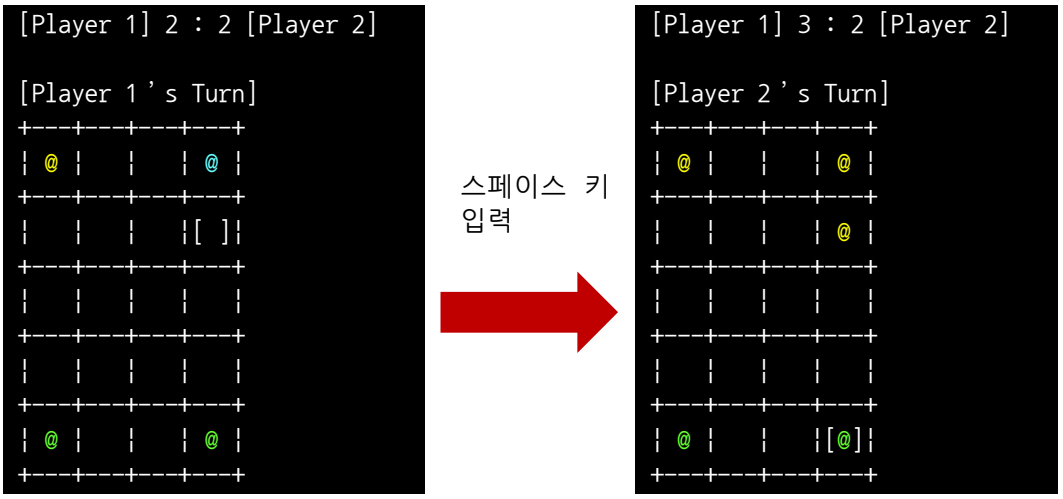


그림 4-1. 말 선택 후 커서 이동　　그림 4-2. 말이 복제된 화면

그림 4. 말 선택 및 복제 예시

(5) 다음 플레이어의 선택 및 이동

한 플레이어가 자신의 말을 복제(또는 이동)하고 나면, 상대방 플레이어의 순서가 된다. 그림 4-2를 보면 Player 2의 순서가 된 것을 볼 수 있다. 그리고, 플레이어 2의 처음 커서 위치는 (row-1, col-1)로 여기에서는 (4, 3)의 위치에서 시작한다.

플레이어 2는 그림 4-2에서 스페이스를 눌러 이동할 말을 선택한다. 그림 5-1을 보면, 선택된 말이 하늘색으로 표시된 것을 볼 수 있다.

그림 5-2와 5-3은 말 이동을 위하여 i(위)키를 눌러 이동하는 화면을 순서대로 나타내고 있다. (2, 3)의 위치는 옮길 말의 위치에서 두 칸 차이가 나기 때문에 말 이동이 가능한

위치이다. 그림 5-3 화면의 상황에서 플레이어 2가 스페이스를 눌러 (4, 3)의 위치에 있던 말이 (2, 3)의 위치로 이동했음을 그림 5-4에서 볼 수 있다. 그리고 규칙에 따라 말을 둔 곳의 인접한 지점의 말 (1, 3)에 해당하는 플레이어 1의 말이 플레이어 2의 말이 되었음을 볼 수 있다.

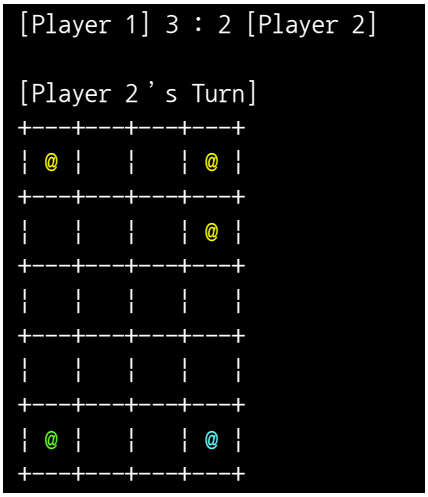


그림 5-1. 말 선택

i(위)키 입력

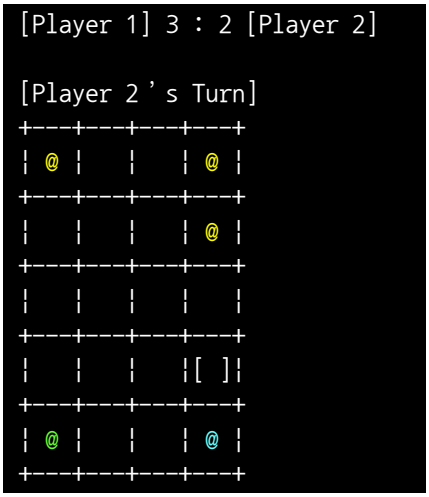


그림 5-2. 말 선택 후 커서 이동

i(위)키 입력

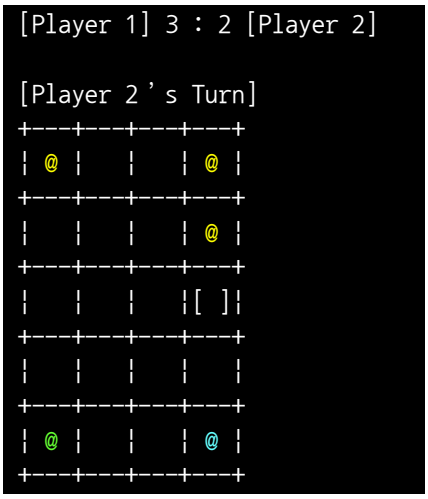


그림 5-3. 말 선택 후 커서 이동

스페이스 키 입력

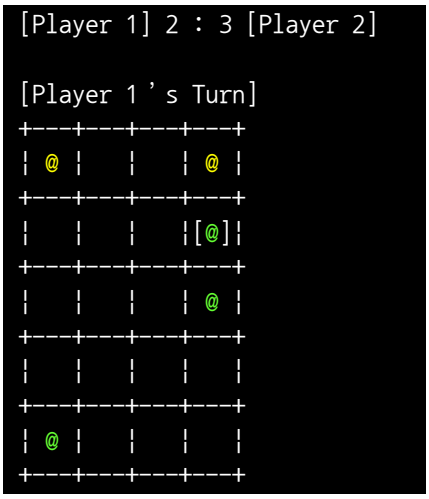
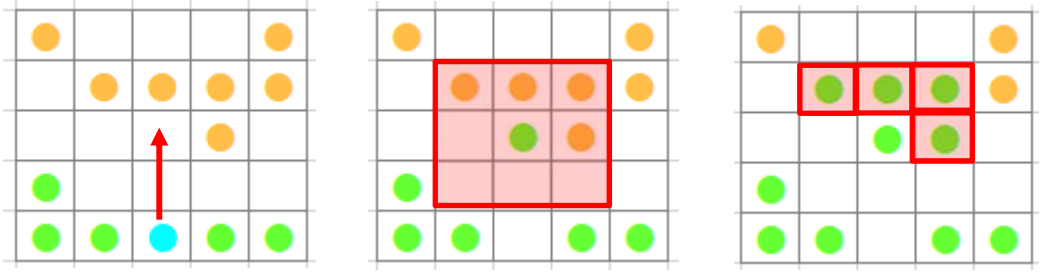


그림 5-4. 말이 이동된 화면

그림 5. 말 선택 및 이동 예시

(6) 플레이어가 말을 두게 되면, 둔 곳으로부터 인접한 지점(가로, 세로, 대각선으로 한칸 떨어진 곳)에 상대방의 말이 있을 경우 그 말은 모두 자신의 말로 바뀐다.



(a) (b) (c)

그림 6. 말 이동 후, 인접한 지점에 대한 예시

그림 6을 보면 연두색 플레이어의 말이 선택(하늘색 말)되어 화살표 끝으로 이동을 했다(그림 6. (b)). 이동한 곳의 인접한 곳을 빨간색 네모로 표시를 했다. 이 안에 들어 있는 상대방 말은 다 연두색으로 변경되었음을 볼 수 있다(그림 6. (c)).

(7) 게임 종료

한 플레이어가 자신의 말을 이동(또는 복제)하고 나면, 상대방 플레이어가 자신의 말을 선택하여 이동(또는 복제)하게 된다. 이렇게 순서를 번갈아가면서 게임을 진행하다가, 두 플레이어 모두 더 이상 둘 곳이 없으면 게임이 끝나게 된다.

- 게임 종료 조건
  - 말이 게임 판 전체에 가득찬 경우
  - 어느 한쪽 플레이어의 말만 남은 경우

게임 종료 전, 만일 한쪽 플레이어가 상대방을 완전히 포위해서 상대방이 더 이상 둘곳이 없다면, 나머지 칸을 다 채울 때까지 혼자서 게임을 진행한다.

아래 그림 7은 게임 종료 화면으로 Player 2가 11대 9로 승리한 화면이다. 게임 종료 시, 말이 차지하고 있는 최종 숫자를 비교하여 어떤 플레이어가 이겼는지 판단하여 해당 출력 메시지를 출력한다. ('Player 1 wins', 'Player 2 wins', 'Draw' 중 출력)

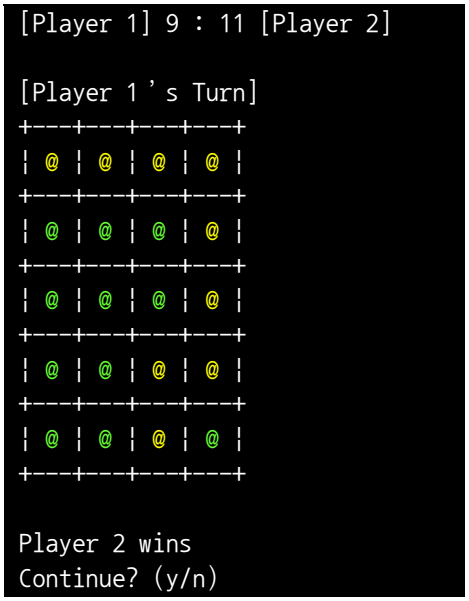


그림 7. 게임 종료 – Player 2 승리 화면

(8) 게임 다시 시작 및 프로그램 종료

그림 7과 같이 게임이 종료 되면, 게임을 계속 진행할지 여부를 질문 받는다.

- 'y' 입력 시, 게임 판 생성을 위해 동적할당 받은 메모리를 반드시 모두 해제 후, 새 게임 판을 만들기 위해 그림 1의 화면이 출력되어 행과 열 크기를 입력받는 부분부터 다시 시작한다.
- 'n' 입력 시, 동적할당 받은 모든 메모리를 반드시 해제 후 프로그램을 종료한다.

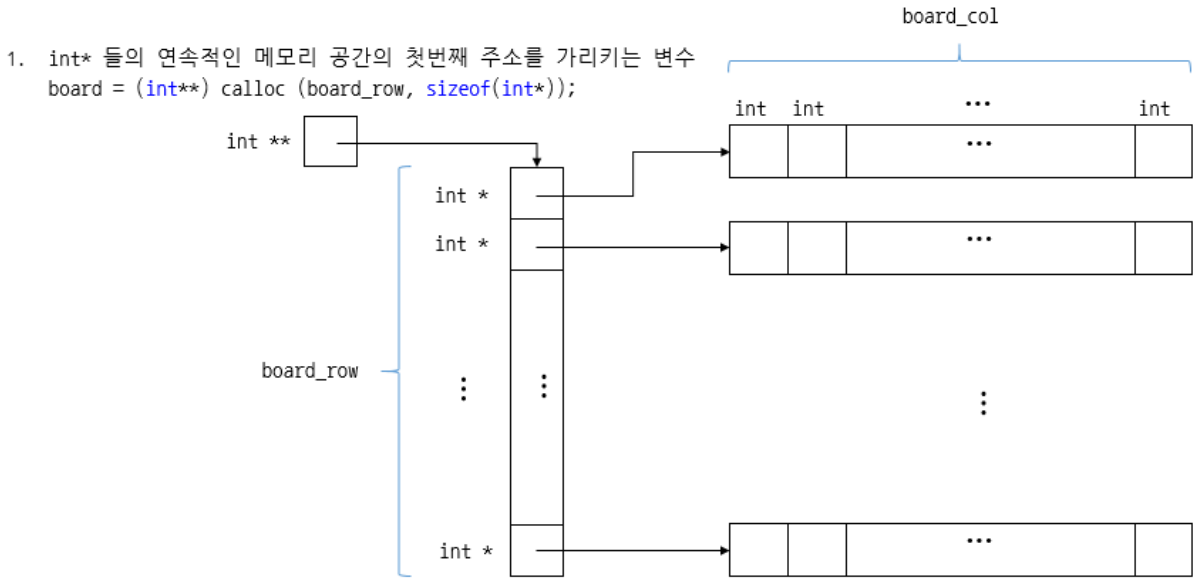
(힌트)

- **게임 판을 생성하기 위한 2차원 배열 동적 할당 방법**
  - 게임 판 사이즈는 변경 될 수 있기 때문에 동적할당을 이용한다.
    int **board는 2차원 배열을 동적할당 받기 위한 포인터 변수이다.

```
int **board;

// 2차원 int 배열 메모리 할당
board = (int **)calloc(board_row, sizeof(int *));
for(i = 0; i < board_row; i++)
        *(board + i) = (int *)calloc(board_col, sizeof(int));
```

  - **2차원 배열에 대한 모식도**



2. 각각의 int* 는 int 들의 연속적인 메모리 공간의 첫번째 주소를 가리키는 변수

```
for(i = 0; i < board_row; i++)
     *(board + i) = (int *)calloc(board_col, sizeof(int));
```

- **엔터를 누르지 않고도 키보드 입력 받기**

int _getch(void): 화면에 출력 없이 키보드로부터 하나의 문자를 입력 받는 함수

  - 엔터를 누르지 않고도 키보드 입력을 받을 수 있고, 누른 키가 화면에 출력되지 않기 때문에 게임을 엔터키 없이 진행할 수 있다.
  - 누른 키보드의 값을 리턴한다.
  - <conio.h>에 포함되어 있음

- **글자색 출력**

<Windows.h>를 포함한 뒤, 아래 코드와 같이 SetConsoleTextAttribute() 함수를 실행해보고 색 지정이 어떻게 이뤄지는지 파악한다.

```
#include <stdio.h>
#include <Windows.h>
int main()
{
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11); // 연한 하늘색
	printf("이 텍스트는 빨간색입니다. \n");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10); // 연한 초록색
	printf("이 텍스트는 초록색입니다. \n");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 14); // 연한 노란색
	printf("이 텍스트는 노란색입니다. \n");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 7);  // 흰색
	printf("이 텍스트는 흰색입니다. \n");

	return 0;
}
```