CSED101. Programming & Problem solving

Spring, 2021

Programming Assignment #3

(75 points)

김진수 (fusion4268@postech.ac.kr)

■ **제출 마감일: 2021.05.20 23:59**

■ **개발 환경: Windows Visual Studio 2019**

■ **제출물**

- **C Code files (`mystring.h, mystring.c, 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 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 각 문제에 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
- 컴파일 & 실행이 안되면 무조건 0점 처리된다
- 추가 기능 구현에 대한 추가 점수는 없다.
- **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공 학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)

[들어가기 전]

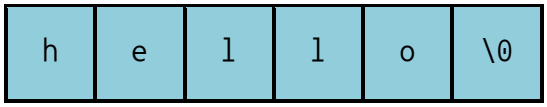
1. 문자열(string)

- 연속된 문자들로 C 언어에서 문자열 앞 뒤에 " "를 이용한다.
- char 형의 1차원 배열을 이용하여 문자열을 저장한다.
- 배열에 문자열을 저장할 때는 끝에 NULL 문자 ('\0')를 넣어서 표시한다.

2. 선언

- **char str[] = "hello";**
  위와 같이 선언과 동시에 초기화를 하게 되면, 자동으로 문자열의 끝에 NULL 문자가 추가된다.

  str



3. 입력과 출력

```
char str[100];
printf("Input your favorite color: ");
scanf("%s", str);
printf("%s", str);
```

<실행 예시> (아래의 빨간색 밑줄은 사용자 입력에 해당)

```
Input your favorite color: blue
blue
```

■ Problem 1: 문자열 처리 함수 (3점)

(문제)

C 언어에서 제공되는 문자열 함수는 <string.h> 라이브러리에서 제공된다. 문자열 라이브러리에 포함되어 있는 함수 중 일부를 직접 작성해 보자.

(설명)

제공된 assn3_p1.zip 의 압축을 풀면 mystring.h 와 mystring.c 가 주어진다.
mystring.h 는 아래와 같은 문자열 처리 함수의 선언을 포함하고 있다. (그대로 사용하고 변경하지 말 것)

```
#pragma once

int mystrlen(char *str);
char *mystrcpy(char *toStr, char *fromStr);
int mystrcmp(char *str1, char *str2);
```

위 함수의 구현부를 포함한 mystring.c 를 작성하라. 각 함수의 정의는 다음과 같다.

(1) int mystrlen(char *str): 문자열 라이브러리의 strlen() 함수와 동일한 기능을 수행
NULL 문자를 제외한 문자열의 길이를 반환한다. 빈 문자열의 경우 0 을 반환한다.
예제)

```
printf("%d\n", mystrlen("cs101")); // 결과: 5
```

(2) char *mystrcpy(char *toStr, char *fromStr): 문자열 라이브러리의 strcpy() 함수와 동일한 기능 수행
문자열 복사 함수로 NULL 문자를 포함한 문자열 fromStr 을 문자열 toStr 에 복사한 후, 문자열 toStr 의 시작 주소를 반환한다.
예제)

```
char str[256];
printf("%s\n", mystrcpy(str, "Good Day")); // 결과: Good Day
printf("%s\n", mystrcpy(str, "Hello"));     // 결과: Hello
```

(3) int mystrcmp(char *str1, char *str2): 문자열 라이브러리의 strcmp() 함수와 동일한 기능 수행
문자열 str1 과 str2 의 대소를 비교한다(대소문자 구분). 비교 기준은 아스키코드표의 값을 기준으로 한다.
각 문자열의 첫 번째 문자부터 비교를 시작한다. 만일 문자가 같다면 두 문자가 다를 때까지 또는 NULL 에 도달할 때까지 계속 비교한다.

- 비교 중 str1 의 문자가 작을 경우 -1, 클 경우 1 을 반환한다.
- 문자열이 길이가 같고 모든 문자가 같을 경우, 0 을 반환한다.
- 비교 중 하나의 문자열이 먼저 끝에 도달할 경우, 먼저 끝난 문자열을 작다고 판단한다.

예제)

```
printf("%d\n", mystrcmp("csed101", "csed103")); // 결과: -1
printf("%d\n", mystrcmp("csed", "Csed"));        // 결과: 1
printf("%d\n", mystrcmp("csed", "cse"));         // 결과: 1
printf("%d\n", mystrcmp("csed", "csed103"));    // 결과: -1
```

■ Problem2: 빙고 게임 2 (72점)

[문제]

Assn2에서는 2차원 배열을 이용하여 알파벳을 원소로 가지는 빙고게임을 구현하였다. 본 과제에서는 단어를 원소로 가지는 2차원 배열을 동적 할당 받아 빙고게임을 구현하도록 한다. 컴퓨터는 단어를 무작위로 선택하는 것이 아닌 '우선 순위 보드'의 규칙에 따라 단어를 선택해야한다.

[목적]

본 과제는 다음의 내용을 목적으로 한다.

- string의 사용법을 익힌다.
- 포인터의 동적 할당과 해제 사용법을 익힌다.
- 파일 입출력의 사용법을 익힌다.
- 우선 순위 보드를 구현함으로써 알고리즘 구현 능력을 기른다.

[주의사항]

- 파일 이름은 "assn3.c"로 저장 한다.
- 보고서 이름은 "assn3.docx", "assn3.hwp" 또는 "assn3.pdf"로 저장한다.
- 전역 변수, goto 문, 구조체, 객체지향프로그래밍의 개념은 사용할 수 없다.
- 문제의 출력 형식은 채점을 위해 아래의 실행 예시와 최대한 유사하게 출력한다.
- 프로그램에서 랜덤 시드는 프로그램 시작 시 main() 에서 srand(time(NULL)); 함수를 한번만 호출하도록 하여 한번만 초기화 한다.
- 명시된 예외 처리 외에는 고려하지 않아도 된다.
- 표준 헤더 파일 <string.h>를 include하여 사용할 수 있다.
- 모든 기능을 main 함수에서 구현한 경우 감점 처리한다.
  - 기능적으로 독립됐거나 반복적으로 사용되는 기능은 사용자 함수를 정의해서 구현한다.
- 프로그램 구현 시, main()함수를 호출하여 사용하지 않는다.
- 프로그램 시작 시, 아래와 같이 main()함수에서 사용자 빙고 보드 포인터, 컴퓨터 빙고 보드 포인터, 우선 순위 보드 포인터를 선언한 후, 시작하도록 한다.

```
char*** user_board = NULL;     //사용자의 빙고 보드
char*** comp_board = NULL;     //컴퓨터의 빙고 보드
int** priority_board = NULL;   //컴퓨터의 우선 순위 보드
```

  빙고 보드는 반드시 3중 포인터를, 우선 순위 보드는 2중 포인터를 사용하여 동적으로 할당한 뒤 사용 및 해제 한다(8쪽, 11쪽 참고).

  - 동적 할당을 이용하지 않고 크기가 정해진 배열을 선언 후 사용 할 시 0점 처리한다.

[함수 요구 사항]

다음의 사용자 정의 함수들을 사용하여 구현하도록 한다. 아래 기능 외의 필요한 함수를 정의해 사용할 수 있다.

- `#define MAX_WORD_LENGTH 9`
  - 단어의 최대 알파벳 수를 9개로 제한한다.
- `void userTurn(char*** user_board, char*** comp_board, …)`
  - 사용자의 턴에 게임 진행 (단어 입력 등)

- `void compTurn(char*** user_board, char*** comp_board, …)`
  - 컴퓨터의 턴에 게임 진행 (우선 순위 보드로부터 단어 선택)
- `int countBingo(char*** board, …)`
  - 빙고 보드(board)에서 빙고 개수를 세어서 반환
- `void updatePriority(int** priority_board, …)`
  - 컴퓨터의 빙고 보드에 변화가 있을 때 우선 순위 보드를 업데이트 하는 함수
- `void  freeBingo(char*** user_board, char*** comp_board, int** priority_board, …)`
  - 게임이 종료되면, 동적 할당 받은 빙고 보드 (사용자, 컴퓨터)와 우선 순위 보드를 할당 해제

[게임 진행 설명]

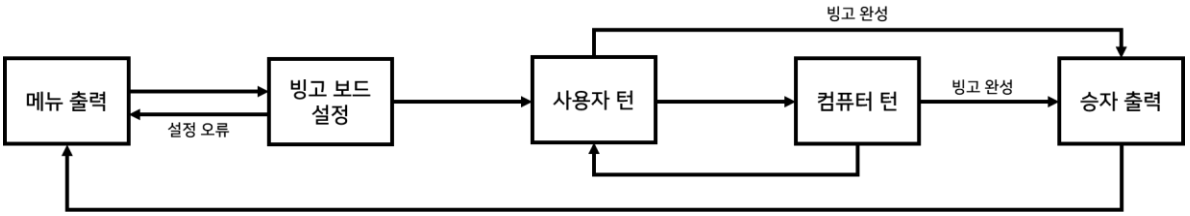


그림 1. 본 게임의 흐름도

게임은 다음과 같은 순서로 진행된다. 사용자가 메뉴 화면에서 시작을 선택하면 빙고 보드를 설정한다. 입력한 값이 빙고 보드를 생성하는 데 적합하지 않은 경우 오류를 표시하고 다시 메뉴를 출력한다. 사용자의 턴이나 컴퓨터의 턴에 빙고가 완성되면 승자를 출력한다. 승자가 출력되어 게임이 종료된 이후 다시 메뉴 화면을 출력한다.

[1. 메뉴 출력]

프로그램이 실행되면 게임 시작을 위한 메뉴를 출력한다. 메뉴 출력 화면은 다음과 같다.

```
[Bingo Game]
==============================
1: Start
2: Exit
==============================
Choice:
```

그림 2. 메뉴 화면

- 1을 입력할 시 빙고 게임을 시작한다.
- 2를 입력할 시 프로그램을 종료한다.
- 1, 2 이외의 숫자를 입력했을 경우, 에러 메시지를 출력하고 새로운 숫자를 입력받는다.
- 정수 외의 입력 값에 대해서는 고려하지 않는다.

```
Choice: 3
Err: Wrong choice.
Choice:
```

그림 3. 1, 2 이외의 숫자를 입력한 경우

[2. 빙고 보드 설정]

본 단계에서는 사용자와 컴퓨터의 빙고 보드를 생성한다. 빙고 보드는 파일로부터 단어를 읽어서 구성하도록 한다. 빙고 보드 생성 단계는 **1) 파일 선택, 2) 빙고 보드 크기 설정, 3) 승리 빙고 개수 설정, 4) 빙고 보드 생성, 5) 빙고 보드 원소 채우기**로 이루어져있다.

1) 파일 선택

- 메뉴 화면에서 1을 입력 했을 때, 사용자로부터 파일 이름을 입력 받아, 해당 파일을 읽도록 한다. (실행 예시의 빨간색 밑줄은 사용자 입력을 나타냄)
- 아래의 실행 예시는, 사용자가 입력한 color.txt 파일을 읽어 그 내용을 화면에 출력한 화면이다.

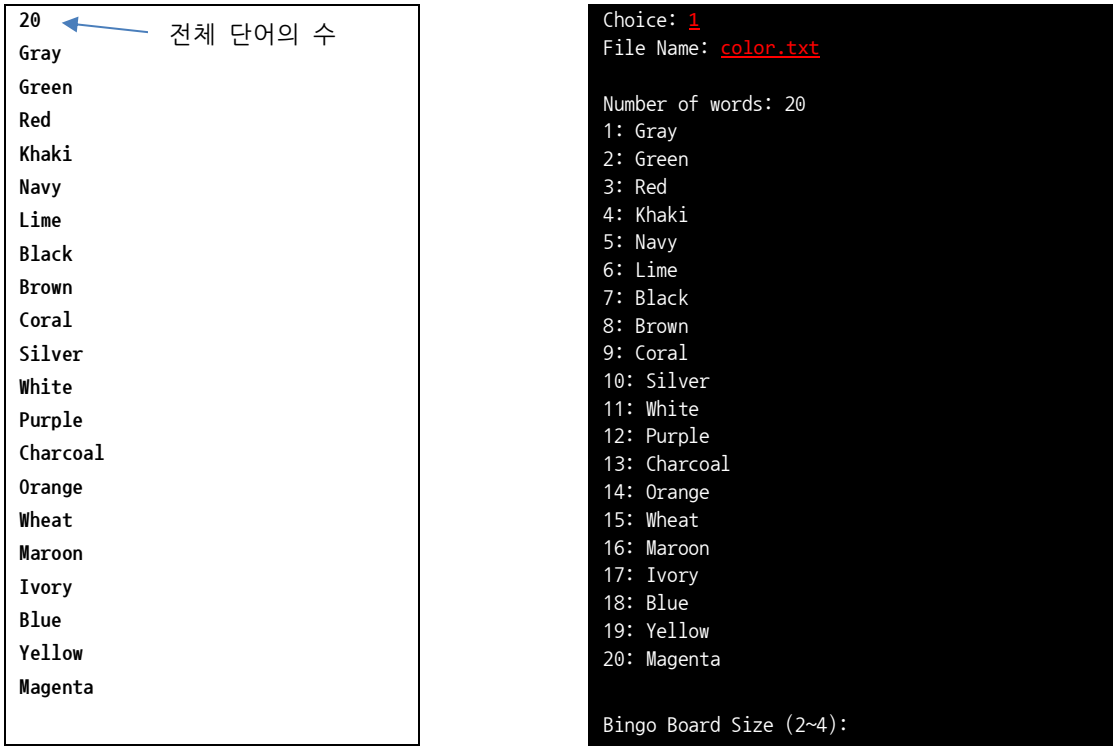


그림 4. 텍스트 파일 예시(왼쪽)와 파일 선택 화면(오른쪽)

- color.txt의 파일의 내용 및 구성은 그림4의 왼쪽과 같다.
  - 첫 번째 줄의 숫자는 전체 단어의 수를 나타낸다.
  - 나머지 데이터들은 영문 단어로 줄(line) 단위로 구분된다.
  - 단어의 최대 길이는 9 라고 가정한다.
  - 파일의 내용이 틀린 경우는 없다고 가정한다. (전체 단어의 수와 실제 단어가 수가 다른 경우, 단어의 길이가 9를 넘는 경우, 최대 단어의 수가 4보다 작은 경우 등은 없다고 가정한다.)
- 파일이 없는 경우 에러 메시지를 출력하고 메뉴 선택 화면으로 돌아간다.

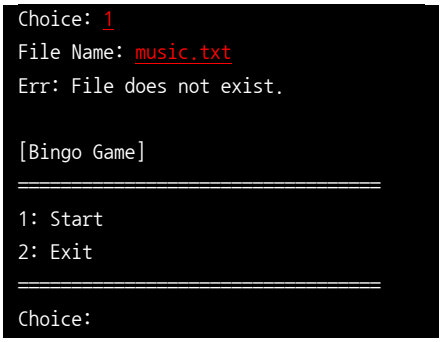


그림 5. 존재하지 않는 파일을 입력한 경우

- 예시로 color.txt, animal.txt, food.txt가 제공되며 같은 양식의 다른 파일에 대해서도 프로그램이 동작해야 한다.
- 파일 이름을 저장할 배열은 아래와 같이 선언하고 사용한다.
  참고로, 실제 채점을 위한 테스트 예제 입력 시, 20자 이내의 파일 이름을 입력하여 테스트할 예정이다.

```
#define MAX_FILE_NAME 30
char filename[MAX_FILE_NAME];
```

2) 빙고 보드 크기 설정

- 빙고 보드의 크기 범위는 선택한 파일의 단어 개수에 따라 결정된다.
- 파일의 단어 개수가 N개라고 할 때, 빙고 보드의 크기는 1보다 크고 $\sqrt{N}$보다 작거나 같은 정수가 되어야 한다.
- 선택한 파일이 color.txt일 때, 빙고 보드의 크기를 묻는 화면은 그림 6과 같다.
- 보드의 크기가 범위를 벗어난 경우, 에러 메시지를 출력하고 메뉴 선택화면으로 돌아간다.
- 정수 외의 입력 값에 대해서는 고려하지 않는다.

```
Bingo Board Size (2~4):
```

그림 6. 빙고 보드 크기 선택 화면

```
Bingo Board Size (2~4): 5
Err: Wrong bingo board size.
```

그림 7. 보드의 크기가 범위를 벗어난 경우

3) 승리 빙고 개수 설정

- 파일과 빙고 보드의 크기가 선택되었다면, 승리를 위한 빙고 개수를 설정해야 한다.
- 빙고 보드의 크기가 S라고 할 때, 승리를 위한 빙고 개수는 최소 1개에서 최대 2S+2(가로 S개, 세로 S개, 대각선 2개)개까지 설정할 수 있어야 한다.
- 선택한 빙고의 크기가 4일 때, 승리 빙고 개수를 묻는 화면은 그림 8과 같다.
- 승리 빙고 개수가 범위를 벗어난 경우, 에러 메시지를 출력하고 메뉴 선택화면으로 돌아간다.
- 정수 외의 입력 값에 대해서는 고려하지 않는다.

```
Bingo Board Size (2~4): 4
Number of bingos to win (1~10):
```

그림 8. 승리 빙고 개수 선택 화면

```
Bingo Board Size (2~4): 4
Number of bingos to win (1~10): 13
Err: Wrong numbers to win.
```

그림 9. 승리 빙고 개수가 범위를 벗어난 경우

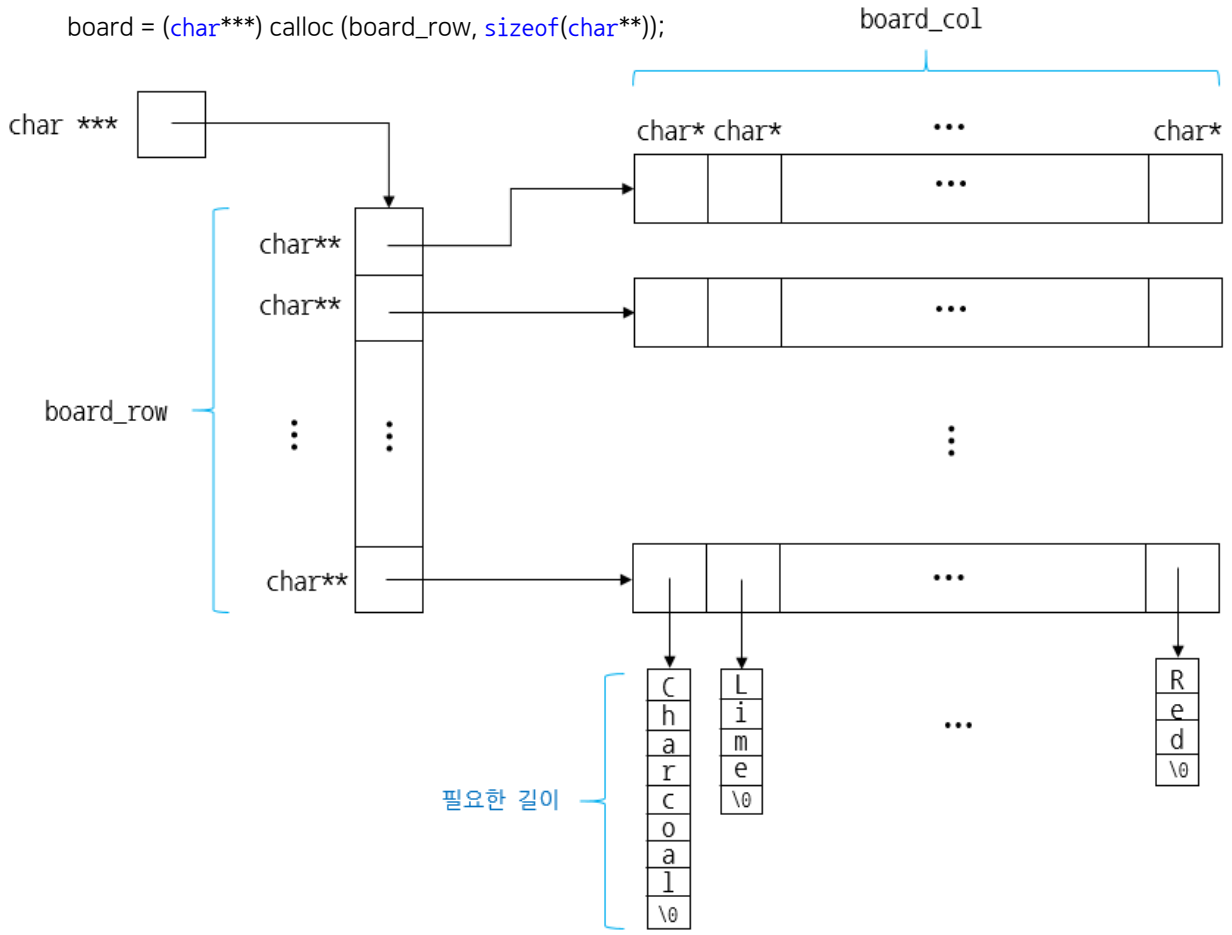
4) 빙고 보드 생성

- 사용자 보드와 컴퓨터 보드를 각각 동적 할당 받아 생성한다.
- 빙고 보드 사이즈는 매 게임마다 변경될 수 있기 때문에 동적 할당을 이용한다.
  char pointer를 원소로 가지는 2차원 배열을 동적할당해야 한다.

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

- 2차원 배열에 대한 모식도

(1) char** 들의 연속적인 메모리 공간의 첫번째 주소를 가리키는 변수
board = (char***) calloc (board_row, sizeof(char**));



(2) 각각의 char**는 char* 들의 연속적인 메모리 공간의 첫번째 주소를 가리키는 변수

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

(3) 각각의 char *들은 문자열의 시작 주소를 가리키는 변수로 필요한 길이만큼 동적 할당 받아 해당 단어를 저장하도록 한다.

```
board[board_row-1][0] = (char *)malloc(sizeof(char) * (문자열 길이 + 1));
```

5) 빙고 보드 원소 채우기

- 빙고 보드 생성 방식은 shuffle을 한 경우와 shuffle을 하지 않은 경우로 나뉜다.
- 0, 1 이외의 입력에 대해서는 고려하지 않는다.

```
Number of bingos to win (1~10): 2
Shuffle (1: Yes, 0: No):
```

그림 10. 빙고 보드 생성 방식 선택 화면

A) Shuffle

- 파일의 단어 개수가 N개이고 빙고 보드의 크기가 S일 때, shuffle을 하였을 경우 N개의 단어 중 S^2개를 임의로 추출하여 빙고 보드를 채운다. (하나의 보드를 채울 때, 중복 단어를 선택할 수 없다.)
- 빙고 보드에 단어를 채울 때, 각 단어마다 필요한 길이만큼 동적 할당 받아 해당 단어를 저장하도록 한다. (2차원 배열에 대한 모식도 참고)
  아래는 파일에서 읽은 문자열 길이만큼 메모리를 동적 할당한 예제이다. 문자열을 읽어올 때, 입력 받는 길이는 미리 알 수 없기 때문에 충분한 크기의 배열(str)을 선언하여 그 곳에 저장하고, 입력 받은 문자열의 길이만큼만 메모리를 동적 할당하여 사용하여야 한다.

```
#define MAX_STR_LEN 30
…
char str[MAX_STR_LEN];  // 문자열을 입력 받을 배열
char *p;                // 동적 메모리 할당을 받을 포인터
fscanf(fp, "%s", str);  // “hello”를 파일에서 읽은 경우, “hello\0”를 저장하는 공간을
                        // 제외한 나머지 메모리가 낭비
p = (char *)malloc(sizeof(char) * (strlen(str)+1)); // str에 저장된 문자열의 크기와 동일한 공간 할당
strcpy(p, str);         // p가 가리키는 공간에 문자열 복사 기능 수행
…
free(p);                // p에 대한 사용이 모두 끝나면, 메모리 할당 해제
```

- 사용자와 컴퓨터의 빙고 보드 각각에 대해서 위의 방식으로 랜덤하게 채운다.
- color.txt 파일로 보드의 크기가 4인 shuffle 방식 빙고 보드를 생성하였을 때, 사용자와 컴퓨터의 빙고 보드는 그림 11과 같다. 왼쪽이 사용자의 빙고 보드이고 오른쪽이 컴퓨터의 빙고 보드이며, 두 빙고 보드 모두 단어가 어디에 배치되어 있는지 출력하도록 한다.

```
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Wheat    |Yellow   |Charcoal |Lime     |              |Silver   |Coral    |Red      |Black    |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|White    |Maroon   |Blue     |Green    |              |Yellow   |Purple   |White    |Navy     |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Black    |Coral    |Orange   |Magenta  |              |Brown    |Blue     |Khaki    |Ivory    |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Purple   |Gray     |Navy     |Khaki    |              |Wheat    |Charcoal |Green    |Magenta  |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
```

그림 11. color.txt 로 생성한 크기가 4인 shuffle 빙고 보드

B) Non-shuffle

- 사용자의 빙고 보드는 파일의 첫번째 단어부터 S^2번째 단어까지 순서대로 빙고 보드에 넣어 생성한다.
- 컴퓨터의 빙고 보드는 파일의 N번째 단어부터 $N-S^2+1$번째 단어까지 파일의 역순으로 빙고 보드에 넣어 생성한다.
- color.txt 파일로 빙고 보드 크기가 4인 non-shuffle 방식 빙고 보드를 생성하였을 때, 사용자와 컴퓨터의 빙고 보드는 다음과 같다.

그림 12. color.txt 로 생성한 크기가 4인 non-shuffle 빙고 보드

빙고 보드가 생성되었다면, 콘솔 화면에 그림 11이나 12와 같이 빙고 보드를 출력한다.
출력된 단어들을 왼쪽 정렬 된 상태여야 한다.

[3. 사용자 턴]

게임은 사용자가 먼저 단어를 입력하면서 시작한다. 첫 번째 사용자 턴 이후부터는 컴퓨터와 번갈아 가면서 단어를 선택한다.
그림 13과 같이 사용자 턴에 사용자로부터 단어를 입력받는다.

- 힌트) 단어의 최대 크기는 9로 제한되어 있으므로, 사용자가 입력할 단어를 저장하기 위해서 적절한 크기의 1차원 char 배열을 선언 후 사용하면 된다. (동적 할당 받을 필요 없음)
- 예외 처리) 사용자가 자신의 빙고 보드에 존재하지 않는 단어를 입력한 경우, 아래와 같은 에러 메시지를 출력 후, 다시 단어를 입력 받는다.

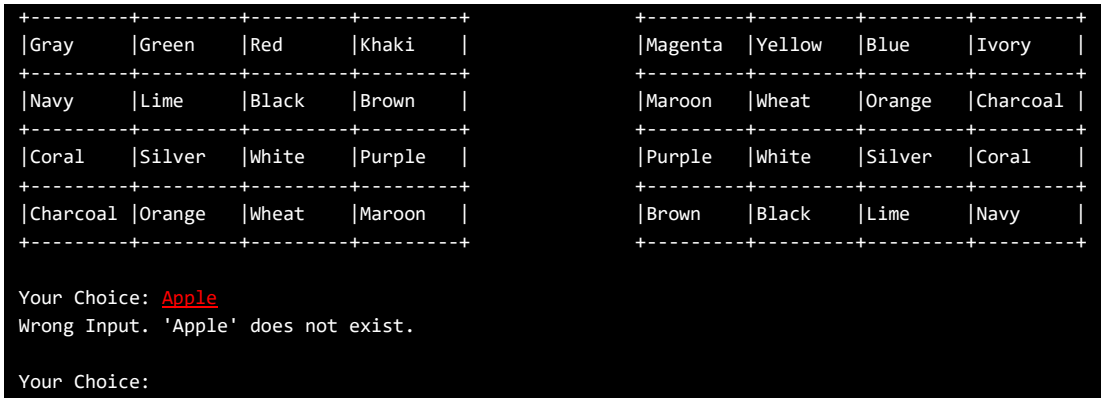


그림 13. 사용자가 빙고 보드에 존재하지 않는 단어를 입력한 경우

자신의 빙고 보드 안에 있는 단어를 입력한 경우, 해당 단어를 그림 14처럼 9개의 '#'이 출력되도록 한다.
해당 단어가 컴퓨터의 빙고 보드에도 존재한다면 컴퓨터의 빙고 보드 또한 해당 단어를 9개의 '#'이 출력되도록 한다.

```
Your Choice: White

+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Gray     |Green    |Red      |Khaki    |              |Magenta  |Yellow   |Blue     |Ivory    |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Navy     |Lime     |Black    |Brown    |              |Maroon   |Wheat    |Orange   |Charcoal |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Coral    |Silver   |#########|Purple   |              |Purple   |#########|Silver   |Coral    |
+---------+---------+---------+---------+              +---------+---------+---------+---------+
|Charcoal |Orange   |Wheat    |Maroon   |              |Brown    |Black    |Lime     |Navy     |
+---------+---------+---------+---------+              +---------+---------+---------+---------+

Computer's Choice:
```

그림 14. 사용자가 'White'를 입력한 경우

사용자가 단어를 선택하고 사용자와 컴퓨터 모두 승리 조건을 충족하지 못한 경우, 컴퓨터의 턴이 된다.

[4. 컴퓨터 턴]

컴퓨터는 컴퓨터 턴에 자동으로 단어를 선택한다. 이 때, 자신의 빙고 보드에서 빙고를 만들기 유리한 위치의 단어를 선택하도록 한다. 본 과제에서는 아래 설명의 '우선 순위 보드' 규칙에 따라 우선 순위가 높은 단어를 선택한다.

1) 우선 순위 보드 생성

우선 순위 보드는 각 단어의 위치에 따라 점수를 부여한 후, 점수가 가장 높은 위치의 단어를 선택하기 위한 보드이다. 따라서 각 단어의 점수를 저장할 2차원 배열이 필요하며, 본 과제에서는 '우선 순위 보드' 를 사용하기 위해 빙고 보드의 크기만큼의 2차원 배열을 동적으로 할당한다.

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

2) 우선 순위 보드 점수 부여

- 우선 순위 보드 생성 시, 각 원소의 값은 해당 원소로 만들 수 있는 빙고의 개수이다.
  빙고 보드의 크기가 3일 때 보드의 좌표는 그림 15와 같다.

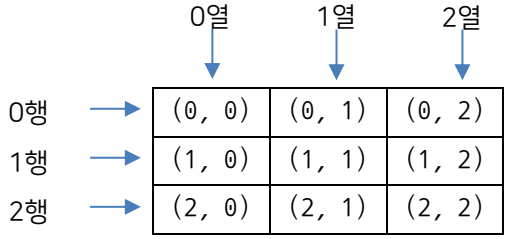


그림 15. 크기가 3인 보드의 좌표

그림 16과 같이 (0,0) 에 있는 원소로 만들 수 있는 빙고의 개수는 3개이고, (0,1) 에 있는 원소로 만들 수 있는 빙고의 개수는 2개이다.

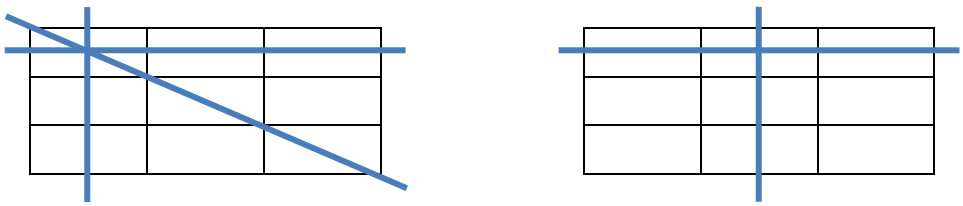


그림 16. (0,0) 에 있는 원소로 만들 수 있는 빙고의 개수 (왼쪽), (0,1) 에 있는 원소로 만들 수 있는 빙고의 개수 (오른쪽)

같은 방법으로 나머지 칸들을 채우면 아래와 같은 우선 순위 보드가 생성된다.

3	2	3
2	4	2
3	2	3

표 1. 빙고 보드의 크기가 3인 우선 순위 보드

- 빙고 보드의 크기가 4일 때 생성된 우선 순위 보드는 다음과 같다.

3	2	2	3
2	3	3	2
2	3	3	2
3	2	2	3

표 2. 빙고 보드의 크기가 4인 우선 순위 보드

3) 단어 선택 규칙

- 생성한 우선 순위 보드는 해당 원소의 값이 클수록 우선 순위가 높음을 나타낸다. 따라서, 컴퓨터는 컴퓨터의 턴에 우선 순위 보드에서 값이 가장 큰 원소를 찾아 해당 위치의 단어를 선택한다.
- 만약 보드 내에 같은 우선 순위 값을 가진 원소가 존재한다면 행이 가장 작은 위치의 단어를 선택한다.
- 같은 우선 순위 값을 가진 원소가 행 또한 같다면 그 중 열이 가장 작은 위치의 단어를 선택한다.

4) 우선 순위 보드 업데이트

사용자의 턴이나 컴퓨터의 턴에 컴퓨터 빙고 보드의 단어가 선택되면 컴퓨터의 우선 순위 보드를 아래의 순서로 업데이트 한다.

가. 선택된 단어가 위치한 우선 순위 보드의 원소를 0으로 설정한다.

나. 선택된 단어와 빙고로 연결된 원소들을 해당 빙고 줄(line)의 0의 개수만큼 증가시킨다.

다음은 사용자가 2행 1열의 원소를 선택한 경우의 예시이다.

가. 선택된 단어가 위치한 우선 순위 보드의 원소를 0으로 설정하고 (표 3 왼쪽),

나. 선택된 위치와 빙고로 연결된 원소들을 해당 빙고 줄의 '0'의 개수만큼 증가시킨다 (표 3 오른쪽).
선택된 칸('0')을 제외한 빙고로 연결된 원소들의 우선 순위 값이 1씩 증가하였음을 확인할 수 있다.

3	2	2	3
2	3	3	2
2	0	3	2
3	2	2	3

3	3	2	4
2	4	4	2
3	0	4	3
4	3	2	3

표 3. 사용자가 2행 1열의 원소를 선택하였을 때의 우선 순위 보드

컴퓨터는 **3) 단어 선택 규칙**에 따라 0행 3열의 원소를 선택하게 된다. 우선 순위 보드 업데이트를 위하여

가. 선택된 단어의 위치에 있는 원소의 값을 0으로 설정하고 (표 4 왼쪽),

나. 선택된 원소와 빙고로 연결된 원소들의 값을 빙고 줄의 '0' 의 개수만큼 증가시킨다 (표 4 오른쪽).

- 하늘색으로 표시한 빙고 줄은, 해당 빙고 줄의 '0'의 개수가 1개이며 해당 줄의 원소의 값이 1씩 증가했다.
- 노란색으로 표시한 빙고 줄은, '0'의 개수가 2개이며 해당 줄의 원소의 값이 2씩 증가했다.
- 단, 값이 0인 원소의 값은 증가하지 않는다.

3	3	2	0
2	4	4	2
3	0	4	3
4	3	2	3

4	4	3	0
2	4	6	3
3	0	4	4
6	3	2	4

표 4. 컴퓨터가 0행 3열의 원소를 선택하였을 때의 우선 순위 보드

사용자가 'White'를 선택하였을 때 컴퓨터는 우선 순위 보드에 따라 자동으로 'Ivory'를 선택하게 되며 실행 화면은 그림 17과 같다.

```
Your Choice: White
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Gray     |Green    |Red      |Khaki    |          |Magenta  |Yellow   |Blue     |Ivory    |
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Navy     |Lime     |Black    |Brown    |          |Maroon   |Wheat    |Orange   |Charcoal |
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Coral    |Silver   |#########|Purple   |          |Purple   |#########|Silver   |Coral    |
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Charcoal |Orange   |Wheat    |Maroon   |          |Brown    |Black    |Lime     |Navy     |
+---------+---------+---------+---------+          +---------+---------+---------+---------+

Computer's Choice: Ivory

+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Gray     |Green    |Red      |Khaki    |          |Magenta  |Yellow   |Blue     |#########|
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Navy     |Lime     |Black    |Brown    |          |Maroon   |Wheat    |Orange   |Charcoal |
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Coral    |Silver   |#########|Purple   |          |Purple   |#########|Silver   |Coral    |
+---------+---------+---------+---------+          +---------+---------+---------+---------+
|Charcoal |Orange   |Wheat    |Maroon   |          |Brown    |Black    |Lime     |Navy     |
+---------+---------+---------+---------+          +---------+---------+---------+---------+

Your Choice:
```

그림 17. 컴퓨터가 'Ivory'를 선택한 경우

컴퓨터가 단어를 선택하고 사용자와 컴퓨터 모두 승리 조건을 충족하지 못한 경우, 사용자 턴이 된다.

[5. 승자 출력]

사용자나 컴퓨터의 빙고 개수가 설정한 승리 빙고 개수보다 크거나 같을 때 게임의 결과를 출력하고 해당 게임을 종료한다.

- 사용자만 완성한 경우: 'You win!'
- 컴퓨터만 완성한 경우: 'Computer wins!'
- 사용자와 컴퓨터 동시에 완성한 경우: 'Draw'

게임이 종료 시, 해당 게임을 위해 동적 할당한 메모리를 모두 할당 해제 한다.

- 빙고 보드(사용자, 컴퓨터)와 우선 순위 보드를 모두 할당 해제할 것

해당 게임이 종료되면 새로운 게임을 시작할 수 있도록 아래와 같이 메뉴를 출력한다.

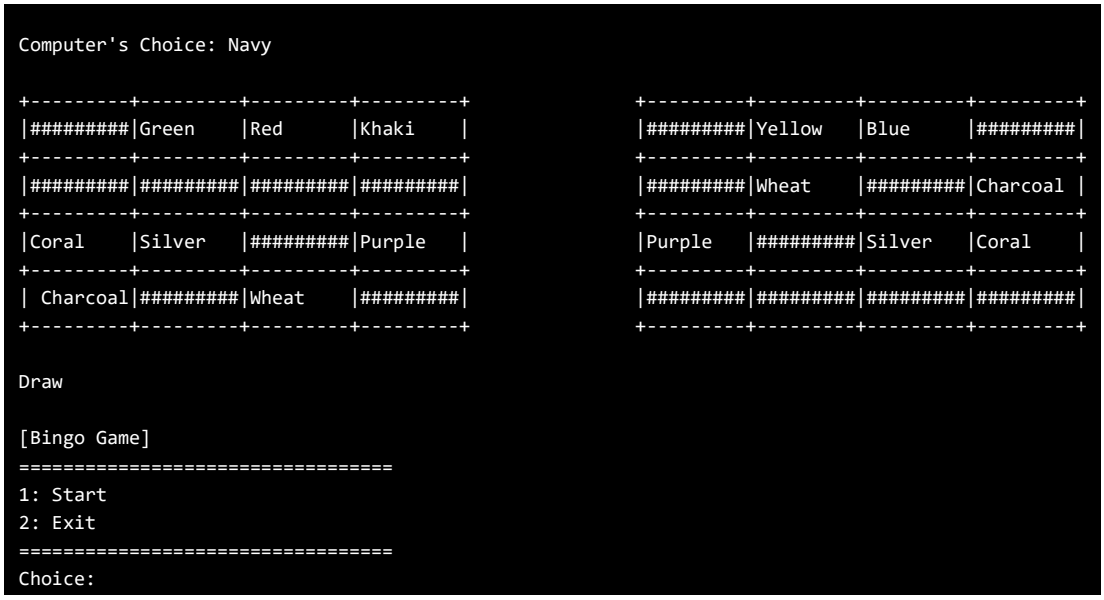


그림 18. 사용자와 컴퓨터가 동시에 2개의 빙고를 완성한 경우

[파일 출력]

파일에는 현재 보드의 상황, 사용자와 컴퓨터의 선택, 컴퓨터의 우선 순위 보드와 게임 결과가 기록되어야 한다.
단, 사용자 입력의 경우, 유효한 입력에 대해서만 기록하면 된다.
파일의 이름은 다음과 같은 규칙을 따른다.

(빙고 보드 크기)(승리 빙고 개수)(Shuffle 여부)_(파일이름).txt

보드 크기와 승리 빙고 개수는 숫자를 사용하며, shuffle하였을 때 1 하지 않았을 때 0을 사용한다.
예시)

- food.txt로 생성한 크기가 3이고 승리 빙고 개수가 3이며 shuffle을 한 경우: 331_food.txt
- animal.txt로 생성한 크기가 4이고 승리 빙고 개수가 1이며 shuffle을 하지 않은 경우: 410_animal.txt
- color.txt로 생성한 크기가 3이고 승리 빙고 개수가 5이며 shuffle을 한 경우: 351_color.txt

(힌트) sprintf() 함수를 사용하여 여러 타입의 변수들을 하나의 string으로 연결할 수 있다.

```
char a[5] = "CSED";
int b = 101;
char c[10];
sprintf(c, "%s_%d", a, b);
printf("%s\n", c);

실행 결과: CSED_101
```

빙고 보드 아래에는 항상 컴퓨터의 '우선 순위 보드' 행렬이 출력되어야 한다. 파일 출력 형식은 assn3_p2.zip의 321_color.txt, 430_animal.txt, 511_food.txt를 참고하여 구현한다.

- 메모장에서 파일 확인 시, 글꼴을 고정폭 글꼴로 변경해 주도록 한다. [서식]->[글꼴] 선택 후, 글꼴을 Consolas로 변경하면 모든 글자가 같은 폭을 갖게 되어, 파일 확인 시 도움이 된다.