

# CSED101. Programming & Problem solving

## Fall, 2015

### Programming Assignment #5 (70 points)

박현준(chun92@postech.ac.kr)

■ **Due:** 2015.12.13 23:59

■ **Development Environment:** Windows Visual Studio 2010

#### ■ **제출물**

- **C Code files (\*.c, \*.h)**

- 이번 과제는 여러 소스 파일로 파일을 분할해서 작성을 한다. 제출시, 모든 파일을 하나의 파일로 압축해서 제출할 것. (압축파일명: **assn5.zip**)
- 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.

- **보고서 파일** (.doc(x) or .hwp) 예) assn5.doc(x) 또는 assn5.hwp

- AssnReadMe.pdf 를 참조하여 작성할 것.
- 프로그램 실행 화면을 캡처하여 보고서에 포함시키고 간단히 설명할 것.

- 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

#### ■ **주의사항**

- 각 문제에 해당하는 요구사항을 반드시 지킬 것.
- 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 과제에서 추가 기능 구현에 대한 점수가 없습니다.
- 이번 과제는 구조체와 연결리스트를 활용을 목표로 함으로 문제에서 지정한 부분에서 구조체와 연결리스트를 사용하지 않고 배열을 통해서 해결할 경우 감점 처리 함.
- 각 소문제에서 지정한 파일에 프로그램을 분할하여 작성하지 않으면 감점 처리 함. (예를 들어 "set.h", "set.c"에 구현해야 할 함수를 "main.c"에 모두 작성하면 감점. 단, 추가적인 파일에 대한 정의는 상관없음.)
- 파일 입력과 출력은 지정한 포맷을 반드시 준수함. 포맷이 맞지 않을 시 감점 처리 함.

## ■ Problem: Freecell 게임

### 1. 개요

프리셀(Freecell)은 조커를 제외한 52 장의 플레이 카드를 사용하여 혼자 주어진 목적을 달성하는 솔리테어(Solitaire)의 일종으로, 마이크로소프트 윈도우에 기본적으로 설치되어 전 세계 많은 사람들이 즐긴 유명한 게임이다. 이번 과제는 주어진 프리셀 문제에 대해 사용자가 직접 프리셀 문제를 풀이할 수 있도록 프리셀 게임을 구현하는 것을 목표로 한다.

### 2. 목적

- 구조체와 연결 리스트의 정의 및 탐색, 삽입, 삭제 등 다양한 활용 방법을 익힌다.
- 소프트웨어 공학의 원리에 따라 프로그램의 모듈에 따라 파일을 분할하고 관리해본다.
- 명령줄인수(argv, argc)의 활용 방법을 익힌다.
- 과제를 통해 프로그램을 작성하고 결과물을 직접 동작하면서 재미와 성취를 느낀다.

### 3. 규칙

#### 3.1. 용어 설명



그림 1. 프리셀 기본 세팅 및 용어 정의

프리셀의 각 용어에 대한 설명은 다음과 같다.

#### (1) 보드

보드는 여덟 개의 줄스택으로 이루어져 있으며 게임이 처음 시작했을 때 52장의 플레이잉 카드가 배치된 공간을 의미한다.

#### (2) 줄스택

게임을 시작할 때 카드가 쌓여있는 공간으로 보드의 한 줄을 의미한다. 플레이어는 줄스택에서 다른 줄스택으로, 프리셀로, 혹은 홈셀로 카드를 이동시킬 수 있다.

#### (3) 홈셀

카드의 무늬 종류에 따라 각각 네 개의 셀이 존재하며, ACE부터 2, 3, 4 순으로 낮은 순으로 카드를 쌓을 수 있는 공간이다. 모든 카드가 홈셀로 이동했을 때 게임이 승리한다.

#### (4) 프리셀

줄스택과는 별개로 임의의 카드를 놓을 수 있는 공간으로 네 개의 셀이 존재한다. 줄스택에서 처리가 어려운 카드를 보관할 수 있다.

### 3.2. 게임의 목표

게임을 시작했을 때 줄스택에 배치된 모든 카드를 주어진 규칙에 따라 홈셀로 이동시키면 게임에서 승리한다. 만약, 이동 가능한 카드가 더 이상 존재하지 않거나 무한 루프(같은 수가 계속 반복됨)에 빠진 경우, 혹은 유저가 문제를 포기할 경우 패배한다.

### 3.3. 세팅

프리셀의 기본 세팅은 다음과 같으며 그림 1 처럼 세팅을 준비한다.

- 조커를 제외한 52 장의 플레이잉 카드 한 벌을 사용한다.
- 카드를 섞는다.
- 8 개의 줄스택에 첫 4 개는 7 장, 나머지 4 개에는 6 장씩 카드를 카드와 숫자가 보이도록 쌓는다.
- 4 개의 프리셀을 만들고, 4 개의 홈셀을 만든다. 처음에 프리셀과 홈셀은 비어있다.

### 3.4. 이동 규칙

기본적으로 한 번에 하나의 카드만 옮길 수 있는 것을 원칙으로 한다. 카드는 줄스택, 프리셀, 홈셀에 규칙에 따라 카드를 자유롭게 옮길 수 있다. 단, 홈셀에 한 번 들어간 카드는 다시 옮길 수 없다.

#### 3.4.1. 홈셀

- 각 셀에 에이스부터 킹까지 순서대로 쌓아 올릴 수 있다. (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K 순) 단, 순서를 빼 먹을 수 없고, 한 셀에는 동일 무늬의 카드만 들어갈 수 있다. (가령, 홈셀 1 번 셀에 스페이드 A 카드가 들어있는 경우, 홈셀 1 번 셀에는 스페이드 2 카드만 옮길 수 있다.)
- 네 개의 셀에 처음 넣을 수 있는 카드는 각 무늬의 A 카드이며, 네 개의 셀에 대해서 어떤 위치에 대하여 무늬의 순서는 자유롭게 넣을 수 있다.
- 한 번 홈셀에 놓은 카드는 홈셀에서 다시 뺄 수 없다.

#### 3.4.2. 줄스택

- 여덟 개의 줄스택에는 기본적으로 카드가 쌓여있으며, 가장 위에 쌓여있는 카드부터 차례로 이동시킬 수 있다. (그림 1 에서 가장 왼쪽의 줄스택의 경우, 클럽 K 카드부터 이동시킬 수 있다. 클럽 K 카드를 움직이기 전까지, 밑에 쌓여있는 다이아몬드 J 카드 이하는 움직일 수 없다.)
- 줄스택에는 카드의 순서를 위에서 아래로 역순으로 카드를 쌓아 올려야 한다. (K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2, A 순), 또한 카드를 쌓을 때는 빨강, 검정, 빨강, 검정 순으로 쌓아 올려야 한다. (그림 1 줄스택 6 의 다이아몬드 Q 카드를 줄스택 1 의 클럽 K 카드 위에 쌓을 수 있다. 마찬가지로, 줄스택 5 의 하트 2 카드를 줄스택 4 의 스페이드 3 카드 위에 쌓을 수 있다. 줄스택 3 의 스페이드 2 카드를 줄스택 4 의 스페이드 3 카드에 쌓는 것은 불가능한데, 두 카드의 색이 같기 때문이다.) 단, 처음에 배치된 순서는 줄스택 이동 규칙과 상관없이 임의로 쌓여있다.
- 하나의 줄스택에 모든 카드가 소비되어 쌓여있는 카드가 없는 경우, 어떤 카드라도 그 자리에 옮길 수 있다.

#### 3.4.3. 프리셀

- 네 개의 셀을 가진 프리셀에는 어떤 임의의 카드라도 임시로 저장할 수 있고, 필요할 때 다시 꺼낼 수 있다.

#### 3.4.4. 프리셀에 따른 줄스택의 카드 여러 장을 하나의 뭉치로 한 번에 이동할 수 있는 규칙

- 카드는 기본적으로 한 번에 한 개씩 이동할 수 있지만, 프리셀과 비어있는 줄스택이 존재함에 따라 줄스택에 쌓여있는 카드 여러 장을 하나의 뭉치로 한 번에 이동할 수 있으며, 한 번에 최대 이동할 수 있는 개수는 다음과 같다.

(1) 이동하고자 하는 위치가 빈 줄스택이 아닌 경우:

(빈 프리셀 개수 + 1) \* (빈 줄스택 개수 + 1)

(2) 이동하고자 하는 위치가 빈 줄스택인 경우:

(빈 프리셀 개수 + 1) \* (빈 줄스택 개수)

- 예제) 가령, 프리셀이 네 개가 모두 비어있고, 빈 줄스택이 하나 존재한다고 하자. 그리고 줄스택 1 번부터 3 번까지 아래의 상태라고 하자.

- 줄스택 1: 10 부터 A 까지 쌓임.
- 줄스택 2: 가장 위에 J 가 쌓임.
- 줄스택 3: 빈 줄스택

(1) 이때, 줄스택 1 에서 줄스택 2 로 옮기고자 하는 경우, 줄스택에 10 부터 A 까지 쌓여있는 덩어리 $((4+1) * (1+1))$ , 총 10 개)를 한 번에 옮길 수 있다. 10 장의 카드뭉치를 움직이는 메커니즘은 다음과 같다.

1) 프리셀에 차례대로 줄스택 1 의 A, 2, 3, 4 를 넣고 줄스택 3 에 5 를 옮긴다.

2) 프리셀 안의 카드를 4, 3, 2, A 순으로 꺼내어 줄스택 3 에 5 위에 쌓는다.

3) 다시 비어있는 프리셀에 줄스택 1 의 6, 7, 8, 9 를 넣는다.

4) 줄스택 1 의 10 을 줄스택 2 에 쌓는다..

5) 프리셀 안의 카드를 9, 8, 7, 6 순서대로 꺼내어 줄스택 2 의 10 위에 쌓는다.

6) 줄스택 3 에서 A, 2, 3, 4 순으로 프리셀에 카드를 넣는다.

7) 줄스택 3 에 5 를 줄스택 2 에 줄스택 2 에 6 위에 쌓는다.

8) 프리셀 안의 카드를 4, 3, 2, A 순서대로 꺼내어 줄스택 2 의 5 위 쌓는다.

9) 줄스택 1 의 10 부터 A 까지의 카드가 줄스택 2 로 전부 이동하였다.

(2) 만약, 줄스택 1 에서 줄스택 3 으로 옮길 경우 최대 5 장 $((4+1)*1)$ 까지 옮길 수 있다.

- 위의 예제에서 알 수 있듯이, 카드를 한 번씩 옮기는 과정을 덩어리로 한 번에 옮김으로써 과정을 생략할 수 있다.

## 4. 구현

프리셀 게임의 구현은 프리셀의 각 모듈에 따라 순차적으로 이루어진다. 4.1과 4.2는 게임 구성을 위한 구조체와 게임 판의 정의, 4.3은 게임 시작 규칙, 4.4는 게임 화면 출력, 4.5는 게임 이동 규칙, 4.6은 승리, 패배 조건 확인에 대한 구현에 대해 기술한다. 4.7은 전체 구현에 대한 흐름도로 4.1부터 4.6까지 각 모듈이 어떻게 구성되고 게임의 흐름에 대해 기술한다. 4.8은 게임의 진행상황을 나타내는 로그 파일 출력에 관하여 기술한다.

### 4.1. 카드 정의 (card.h, card.c)

프리셀 게임에서 사용할 카드에 대한 정의는 **구조체**로 작성한다. 기본적으로 카드 **숫자**와 **무늬**에 대한 정보를 정의한다. 카드의 숫자는 A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K의 13개로 구성되어 있으며, 무늬는 스페이드(Spade), 다이아몬드(Diamond), 하트(Heart), 클럽(Club) 4개로 총 52장의 카드가 존재한다. 또한, 카드 스택(줄스택이나 홈셀)에서 대해 앞, 뒤에 쌓여있는 카드에 접근할 수 있도록(즉, 링크드 리스트를 이용할 수 있도록) 카드 구조체에 대한 포인터를 저장해야 한다. 그 이외에 필요한 정보를 card 구조체에 추가적으로 정의할 수 있다. 아래는 구조체 정의에 대한 예시이다.

```
struct card {
    int number;
    int suit;
    struct card *next;
    ...
};
//card.h
```

카드와 관련된 함수는 자유롭게 작성하되 가능한 card.h에 선언하며, card.c에 정의하도록 한다. (예: 카드의 색을 구하는 함수 등)

### 4.2. 게임 보드 정의 (board.h, board.c)

게임에서 사용하는 줄스택, 프리셀, 홈셀에 대한 정의를 작성한다. 줄스택은 카드의 링크드 리스트로 구성되어 있으며, 줄스택의 헤드에 해당하는 구조체를 정의한다. 홈셀 역시 카드의 링크드 리스트로 홈셀의 헤드에 해당하는 구조체를 정의한다. 그 외에 필요한 정보가 있으면 각 구조체에 추가적으로 선언 할 수 있다. 다음 예제처럼 프리셀 게임은 여덟 개의 줄스택, 네 개의 프리셀, 네 개의 홈셀로 구성되어 있으며 배열로 정의하여 이용할 수 있다.

줄스택, 프리셀, 홈셀에 대한 함수는 자유롭게 작성하되 가능한 board.h에 선언하며, board.c에 정의하도록 한다. (예: 줄스택에 카드 뭉치를 구하는 함수, 선택한 카드에 대해 현재 이동 가능한 줄스택을 구하는 함수 등)

```
struct lineStack {
    struct card *head;
    ...
};
struct homecell {
    struct card *head;
    ...
};
//board.h
```

```
int main() {
    struct lineStack lineStacks[8];
    struct card *freecells[4];
    struct homecell homecells[4];
    ...
}
//main.c
```

### 4.3. 세팅 (set.h, set.c)

프로그램을 처음 시작할 때 카드의 배치를 결정하는 함수를 정의한다. 카드 세팅은 아래와 같이 사용자로부터 명령줄 인수(argv, argc)를 받아들여 결정한다. (노란색으로 표시된 글자는 사용자 입력에 해당)

```
C:\W>assn5.exe file input.txt
```

명령줄 인수는 두 개의 인풋을 가지며 지정하지 않은 형태의 인풋이나 들어오거나 인풋의 개수가 틀린 경우 다음 에러를 화면에 출력하고 종료한다.

```
Error: Invalid command line argument
```

#### 4.3.1. 파일로부터 세팅

프로그램 실행 시, 아래와 같이 입력하면 지정한 파일(input.txt)에서 파일을 읽어 카드를 배치한다.

```
C:\W>assn5.exe file input.txt
```

지정한 파일이 있으면 읽어서 카드 세팅을 수행하고, 없는 경우엔 다음 에러를 화면에 출력하고 종료한다.

```
Error: File doesn't exist
```

인풋 파일은 8행으로 이루어져 있으며 각 행은 다음과 같이 이루어져 있다.

(행번호) (카드1) (카드2) (카드3) ... (카드 n)

행번호는 초기화할 줄스택의 번호고, 카드는 각 줄스택에 쌓일 카드로 왼쪽에 있는 밑에서부터 차례대로 쌓이게 된다. 즉, 카드1이 가장 밑바닥에 놓고, 마지막카드(가장 오른쪽 카드)가 가장 위에 쌓이게 된다.

카드 포맷은 (무늬+카드숫자)로 구성되어 있다.

- 무늬: **S**(스페이드), **D**(다이아몬드), **H**(하트), **C**(클럽)
- 카드숫자: A(1), 2, 3, 4, 5, 6, 7, 8, 9, 10, J(11), Q(12), K(13)

다음은 그림 1에 대한 인풋 파일 예제이다.

```
1 DA D3 DK CJ C5 DJ CK
2 CA H3 H6 D5 C2 D7 D8
3 H4 SQ S5 C5 H10 H8 S2
4 SA CQ D4 C8 HQ C9 S3
5 D2 S8 H9 D9 D6 H2
6 S6 H7 HJ D10 C10 DQ
7 S10 HA S9 HK S4 C4
8 SJ SK C3 C7 S7 H5
```

파일 인풋을 받아들일 때에는 중복되는 카드가 없는지, 52장이 모두 존재하는지, 규칙 3.3에서 언급한 세팅의 규칙 (줄스택 1부터 4까지는 7장, 5부터 8까지는 6장)이 맞는지를 확인하고 맞지 않을 경우 다음과 같은 에러를 출력한다.

```
Error: Invalid input file
```

#### 4.4. 화면 출력(print.h, print.c)

세팅 혹은 카드 이동이 끝날 때마다 콘솔 화면에 현재 게임 진행 상황을 출력한다. 출력은 아래의 그림 2처럼 첫 문단에는 홈셀 정보를, 두 번째 줄에는 프리셀 정보를, 세 번째 줄부터는 줄스택에 대한 정보를 표기한다. 홈셀의 경우, 4 개의 스택을 출력하며, 왼쪽부터 오른쪽으로 카드가 쌓여있는 순서대로 출력한다. 카드 출력은 [무늬+숫자]로 숫자는 2칸을 차지하며 오른쪽정렬을 하도록 한다. 카드와 카드 사이는 스페이스 한 칸을 넣는다. 프리셀의 카드가 없는 경우에는 [ - ] 로 출력하고, 홈셀이 빈 경우에는 각 셀 번호만 출력하며, 줄스택이 빈 경우에는 줄 번호만 출력한다. 카드의 출력 포맷에서 무늬는 하트는 ♥, 다이아몬드는 ◇, 스페이드는 ♠, 클럽은 ♣ 모양을 사용하여 출력하도록 한다.



그림 2 프리셀 출력 예제

```
[Homecell]
1 [♠ A] [♠ 2] [♠ 3] [♠ 4]
2 [♣ A] [♣ 2]
3 [◇ A] [◇ 2] [◇ 3]
4 [♥ A]
[Freecell] [♠ K] [◇ 9] [ - ] [ - ]
[Line Stack]
1 [♣ J] [♥10] [♣ 9] [♥ 8] [♣ 7] [♥ 6] [♣ 5] [◇ 4] [♣ 3]
2
3 [♥ K] [♣ Q] [◇ J]
4 [♥ 5] [♥ 2] [◇ Q] [♥ 7]
5
6 [♣ 8] [♠10] [♣ 6] [♥ 3]
7 [♣ K] [♥ Q] [♠ J] [◇10] [♠ 9] [◇ 8] [♠ 7] [◇ 6] [♠ 5] [♥ 4]
8 [◇ K] [♠ Q] [♥ J] [♣10] [♥ 9] [♠ 8] [◇ 7] [♠ 6] [◇ 5] [♣ 4]

Please enter the target (Line stack - 1, Freecell - 2):
```

## 4.5. 이동(move.h, move.c)

### 4.5.1. 이동할 카드(target) 선택

카드 이동을 위해 먼저 이동할 카드를 선택해야 하므로, 아래와 같이 사용자에게 프리셀과 줄스택 중 어느 곳에서 카드를 옮길지 묻는다. (줄스택이 1, 프리셀이 2)

1) 프리셀을 선택한 경우, 몇 번째 셀을 선택할지를 다시 묻고 선택된 카드를 출력한다.

예제) 그림 2의 첫 번째 프리셀을 선택한 경우

```
Please enter the target (Line stack - 1, Freecell - 2): 2
Please enter the index: 1
Selected Card: [♠ K]
```

- 카드가 존재하지 않는 프리셀을 선택하면 에러를 출력하고 다시 카드를 선택한다.

예제) 그림 2의 세 번째 비어 있는 프리셀을 선택한 경우

```
Please enter the target (Line stack - 1, Freecell - 2): 2
Please enter the index: 3
Error: Illegal card selection
Please enter the target (Line stack - 1, Freecell - 2):
```

2) 줄스택을 선택한 경우, 옮기고 싶은 줄스택과 한 번에 옮길 카드 장수를 입력 받는다.

- 이 때 선택한 카드 개수만큼 카드 뭉치가 이루어지는지 검사해야 한다. 규칙 3.4.4를 만족하는 경우 아래와 같이 선택된 카드를 가장 밑에 쌓여있는 카드부터 순차적으로 출력한다.

예제) 그림 2의 줄스택 1에서 카드 네 장을 선택한 경우

```
Please enter the target (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 1 4
Selected Card: [♥ 6] [♣ 5] [♦ 4] [♣ 3]
```

- 줄스택에 선택한 카드 수만큼의 카드가 없거나 뭉치를 이루지 못하는 경우 다음 예제와 같은 에러를 출력하고 다시 카드를 선택한다.

예제) 그림 2에서 카드 뭉치가 아닌 경우([♦ Q]와 [♥ 7]은 카드 뭉치가 아님)

```
Please enter the target (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 4 2
Error: Illegal card selection
Please enter the target (Line stack - 1, Freecell - 2):
```

### 4.5.2. 옮길 위치(location) 선택

4.5.1에서 성공적으로 카드(target)를 선택한 경우, 그 다음 순서로 옮기고자 하는 위치(location)를 사용자로부터 입력 받아 카드를 옮긴다.

```
Please enter the target (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 1 4
Selected Card: [♥ 6] [♣ 5] [♦ 4] [♣ 3]
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 1
Please enter the index: 2
Moving is successful
```

위의 예제처럼 옮길 수 있는 카드를 성공적으로 선택 한 경우, 다음 순서로 사용자로부터 옮길 위치(줄스택 1번, 프리셀 2번, 홈셀 3번, 취소 4번)를 숫자로 입력 받는다. 위치 선택 후 줄스택, 프리셀, 홈셀에 대해 각각 옮길 칸을 지정한다. 옮길 위치를 이동 가능한 장소로 적절히 선택한 경우에는 카드 이동이 성공하고, 불가능한 위치를 선택한 경우 에러 메시지(Error: Illegal Moving)를 출력하고 카드 선택으로 돌아온다.

1) 옮길 위치 선택에서 **4번 취소를 선택**한 경우에는 이동할 카드 선택(4.5.1.)으로 다시 돌아간다.

```
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 4
Please enter the target (Line stack - 1, Freecell - 2):
```

2) 옮길 위치로 **3번 홈셀**을 선택한 경우, 그 다음 순서로 아래와 같이 사용자로부터 홈셀의 4개의 칸 중 옮길 칸의 번호(1~4)를 입력 받는다.

```
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 3
Please enter the index:
```

(1) 홈셀의 비어있는 칸을 선택 한 경우

- 선택된 카드가 A 카드이면 이동 가능 하고 그 이외의 카드는 이동이 불가능하다.
- 홈셀에는 임의의 순서로 카드 무늬를 배치할 수 있으며, 프리셀과 홈셀은 반드시 1 번 위치부터 채울 필요가 없다. (홈셀 1 이 비어있어도 2 에 카드를 넣을 수 있으며, 원하는 무늬의 카드를 넣을 수 있다.)

(2) 홈셀의 칸에 이미 카드가 있는 경우

- 선택한 칸에 쌓여 있는 마지막 카드와 같은 무늬의 바로 다음 숫자 카드만이 이동 가능하다. 예를 들면, 그림 2 에서 홈셀 2 번 칸에 클럽(♣)무늬의 [♣ A], [♣ 2] 카드가 순서대로 쌓여 있으면, 같은 무늬인 [♣ 3] 카드만이 옮겨질 수 있다.

3) 옮길 위치로 **2번 프리셀**을 선택한 경우, 그 다음 순서로 아래와 같이 사용자로부터 프리셀의 4개의 칸 중 옮길 칸의 번호(1~4)를 입력 받는다.

```
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 2
Please enter the index:
```

(1) 카드 이동 성공

- 아래 예제는 이동 가능한 칸(빈 칸)을 선택한 경우로, 그림 2에서 옮길 카드로 줄스택 4번에서 1장을 선택 한 후, 옮길 위치로 프리셀 3번칸(빈 칸)을 선택하여 카드가 성공적으로 이동된 경우이다.

```
Please enter the target (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 4 1
Selected Card: [♥ 7]
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 2
Please enter the index: 3
Moving is successful
```

- 프리셀에서 옮길 카드(target)를 선택하여 프리셀의 빈 칸으로 이동 할 수 있다. (프리셀 2번칸이 비어있는 경우 프리셀 1번칸의 카드를 프리셀 2번칸으로 이동할 수 있다.)

(2) 카드 이동 실패

- 카드가 이미 채워져 있는 프리셀의 칸을 선택한 경우와 줄스택의 두 장 이상의 카드 뭉치를 프리셀로 이동하는 경우(아래 실행 예제에 해당)는 카드 이동을 할 수 없다. 이 경우 에러 메시지(Error: Illegal Moving)를 출력 후, 다시 카드를 선택한다.

예제) 줄스택의 두 장 이상의 카드 뭉치를 프리셀 3으로 이동하려는 경우

```
Please enter the target (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 1 4
Selected Card: [♥ 6] [♣ 5] [♦ 4] [♣ 3]
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 2
Please enter the index: 3
Error: Illegal Moving
Please enter the target (Line stack - 1, Freecell - 2):
```

- 프리셀의 같은 칸에서 같은 칸으로의 이동은 불가능하다. (프리셀 1번칸의 카드를 프리셀 1번칸으로 이동할 수 없다.)

4) 옮길 위치로 1번 줄스택을 선택한 경우, 그 다음 순서로 아래와 같이 사용자로부터 옮길 줄스택의 번호(1-8)를 입력 받는다.

```
Please enter the location (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 1
Please enter the index:
```

- 선택된 카드(target)는 해당하는 줄스택의 가장 위에 쌓여 있는 카드와 색은 다르고 숫자는 하나 아래의 숫자여야 이동이 가능하다. (3.4.2.의 카드 이동 규칙 참조)
- 선택된 카드가 뭉치인 경우에는 가장 밑에 쌓여 있는 카드가 줄스택의 가장 위에 쌓여있는 카드와 색은 다르고 숫자는 하나 아래이며, 뭉치의 개수가 최대 이동 가능 개수를 초과하지 않아야 이동이 가능하다.
- 아래 예제는 그림 2에서 줄스택 7의 [♣ K]부터 [♥ 4]까지 총 10장의 카드 뭉치를 선택하여 빈 줄스택 2로 옮기고자 할 때, 최대 이동 개수를 초과하여 에러가 발생한 경우이다. (빈 프리셀이 2, 빈 줄스택이 2 이므로 이동 가능한 뭉치의 수는 6장이다. 3.4.4.의 줄 스택의 카드 뭉치 이동 규칙 참조) 이 경우 에러메시지(Error: Illegal Moving)를 출력 후, 다시 카드를 선택한다.

예제) 그림 2에서 카드 뭉치의 수가 초과한 경우

```
Please enter the location (Line stack - 1, Freecell - 2): 1
Please enter the index and a number of cards: 7 10
Selected Card: [♣ K] [♥ Q] [♠ J] [♦ 10] [♠ 9] [♦ 8] [♠ 7] [♦ 6] [♠ 5] [♥ 4]
Please enter the target (Line stack - 1, Freecell - 2, Homecell - 3, Cancel - 4): 1
Please enter the index: 2
Error: Illegal Moving
Please enter the location (Line stack -1, Freecell - 2):
```

※ 예외처리에 대해서

옮길 타겟(1~2)과 위치(1~4)값 입력에 대해서 해당 범위를 넘는 수가 들어오지 않는 것을 가정한다. 또한, 줄스택(1~8)과 프리셀, 홈셀(1~4)의 범위를 넘는 수가 들어오지 않는 것을 가정한다.

#### 4.6. 승리 조건 확인(checkWin.h, checkWin.c)

게임의 승리 조건을 확인하여 결과를 화면에 출력한다. (조건을 만족하는 경우만 출력한다.)

- 승리 조건: 홈셀에 모든 카드가 쌓여 있다.
- 예제) 승리 조건 만족

You Win

#### 4.7. 전체 게임 흐름(main.c)

main.c에는 4.1부터 4.6까지 작성한 모듈을 바탕으로 전체 게임에 대한 흐름을 관리하는 코드를 작성한다. 코드의 흐름은 그림 3의 플로우 차트를 따라 작성한다. 프로그램을 실행시키면 우선 카드를 세팅한다.(4.3.) 세팅에 성공한 경우, 게임 보드(홈셀, 프리셀, 줄스택)를 화면에 출력한다.(4.4.) 그 후 승리 조건을 확인하여 승리가 결정된 경우엔 승리 메시지를 출력하고 게임을 종료한다. (4.6.) 게임이 아직 끝나지 않은 경우 카드를 이동하고 (4.5.) 카드 출력으로 다시 돌아간다.

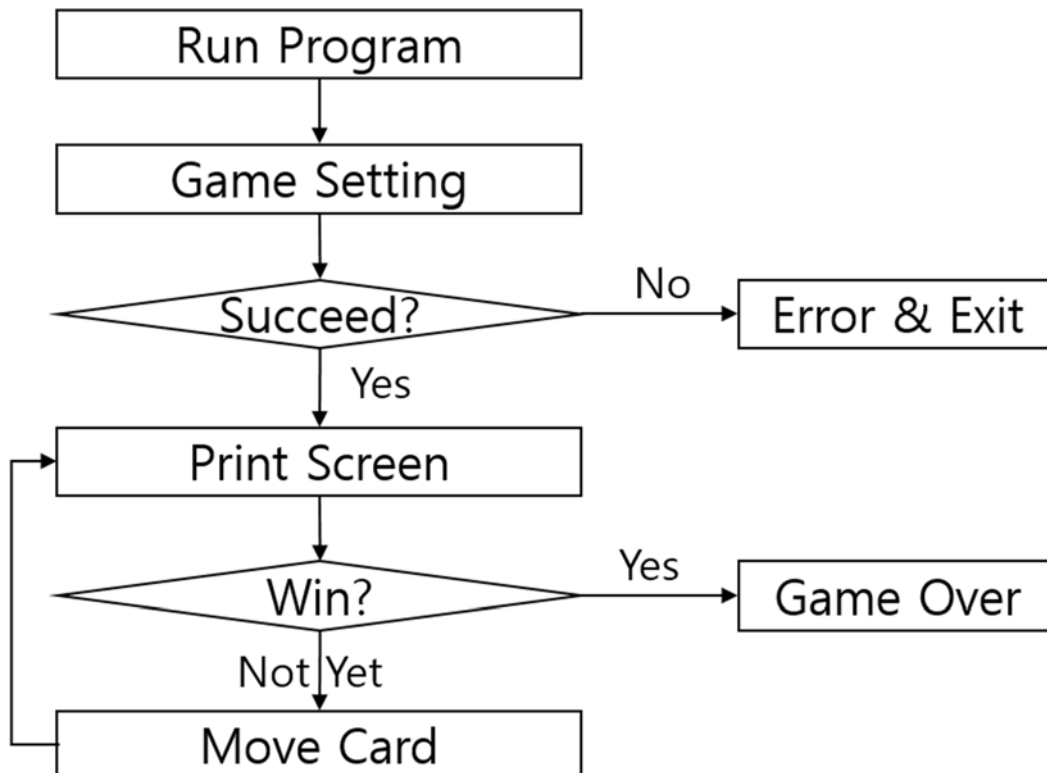


그림 3 게임 전체 흐름 플로우 차트

#### 4.8. 로그(log.h, log.c)

게임 시작, 카드 이동, 게임 종료에 대해서 화면 출력 이외에 게임 로그를 파일 출력을 통해 따로 출력한다. 이는 결과 확인과 채점에 편의성을 위함으로 "log.txt" 파일에 다음과 같은 내용을 저장해야 한다.

##### 1) 게임시작

```
Start
```

##### 2) 게임 종료

예제) 승리

```
Win
```

##### 3) 카드 이동

카드 이동은 이동이 에러 없이 성공한 경우만 기록하며 다음과 같은 포맷으로 기록한다. 카드 포맷은 인풋 파일의 포맷(무늬+카드숫자)으로 작성한다. 이 때 target(옮길 카드), location(위치)에 대해서 줄스택은 1, 프리셀은 2, 홈셀은 3으로 표기한다.

- target(옮길 카드)이 프리셀인 경우

(target) (index) (card) (location) (index)

예제) 프리셀 1번 셀에서 D3를 줄스택 1번으로 이동

```
2 1 D3 1 1
```

- target(옮길 카드)이 줄스택인 경우

(target) (index) (card\_num) [cardlist] (location) (index)

예제) 줄스택 3번에서 C5 H4 S3을 줄스택 2번으로 이동

```
1 3 3 C5 H4 S3 1 2
```

로그 기록에 있어서 카드 이동과 다음 카드 이동 사이의 기록은 엔터를 넣어 구분한다.

어싸인 문서와 함께 제공되는 "5814.txt" 파일과 "log.txt"파일은 예제 프리셀 게임 인풋 파일과 해당 게임에 대한 솔루션 풀이 로그 파일로, 로그 작성시 참조하여 작성한다. 해당 솔루션은 <http://freecellgamesolutions.com/>에서 제공하는 솔루션을 참조하였다.