CSED101. Programming & Problem solving

Fall, 2014

Programming Assignment #5

(70 points)

정진하(jhjeong2014@postech.ac.kr)

- ***Due***: **2014.12.13 23:59**
- ***Development Environment***: **Windows Visual Studio 2010**

- ***제출물***
  - **C Code files (assn5.c)**
    - 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.
  - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- ***주의사항***
  - 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.)
  - assn5.c로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 과제를 작성하는데 있어서 전역변수를 선언하여 사용할 수 없다.
  - main 함수를 포함한 프로그램 내 모든 함수의 길이는 각각 주석을 포함하여 200줄을 넘지 않도록 한다. 즉, 필요한 함수를 적절히 정의하여 활용하도록 한다. 또한 main 함수를 호출해서는 안 된다.
  - stdio.h, stdlib.h, time.h에서 제공하는 라이브러리 외 다른 라이브러리나 수업시간에 배우지 않은 함수를 사용해서는 안 된다.
  - 본 과제에서 추가구현에 대한 점수는 없다.

■ Problem: One Card(원카드)

(문제)

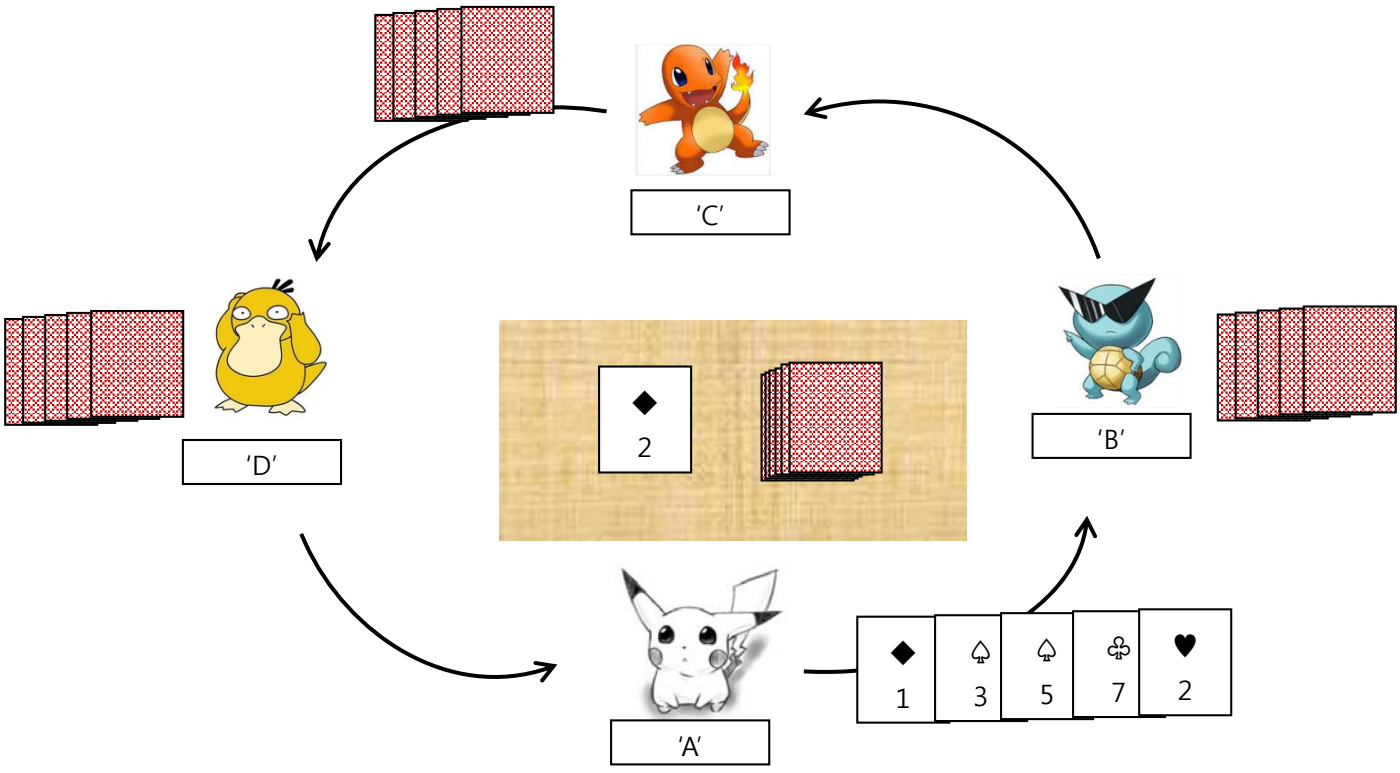
원카드는 조커 카드 2장을 포함한 playing card 54장을 사용하는 카드게임으로 플레이어가 가지고 있는 카드를 먼저 모두 버리는 것을 목적으로 하는 게임이다. 기본적인 규칙은 플레이어가 자기 차례에 테이블에 오픈되어 있는 카드와 같은 무늬의 카드, 또는 같은 숫자나 같은 영문(J, Q, K)의 카드를 한 장씩 내는 것이다. 세부적인 규칙은 지역에 따라 다를 수 있으므로 **반드시 다음의 통일된 규칙을 따르도록 한다**.

I. 플레이어의 수는 2~4명중 고를 수 있다. 처음 플레이어당 카드 수는 5장, 또는 7장이 배분될 수 있으나 본 과제에서는 5장씩 지급되도록 한다. 새로운 게임이 시작될 때 마다 다른 카드가 배분되도록 한다.

II. 플레이어들에게 카드를 5장씩 배분한 후에 한 장의 카드를 오픈한다.

III. 처음 게임을 시작할 때 순서는 시계반대방향(오른쪽)이다. (게임중간에 특수 카드 Q에 의해 방향이 바뀔 수 있다)

플레이어의 이름은 2명인 경우 'A', 'B'로, 4명인 경우 'A, 'B', 'C', 'D'로 정해준다. 게임은 'A' 부터 시작되며 다음 차례는 순서대로 'B', 'C', 'D', ... 가 된다.



IV. 각 플레이어는 자기 차례에 기본적으로 한 장의 카드만 낼 수 있으며, 낼 수 있는 카드는 오픈 카드와 같은 무늬의 카드, 또는 같은 숫자나 같은 영문(J, Q, K)의 카드이다.

- 처음 오픈된 카드가 흑백조커인 경우, ♧ 또는 ♤ 무늬의 카드를 낼 수 있다.
- 처음 오픈된 카드가 컬러조커인 경우, ♥ 또는 ◆ 무늬의 카드를 낼 수 있다
- 조커는 오픈 카드의 무늬, 숫자와 상관없이 언제든 낼 수 있다.

※ **오픈 카드**: 현재 플레이어가 낸 카드로 테이블 상에 무늬와 숫자가 보이게 놓여있는 한장의 카드. 위 그림에서는 [◆ 2]가 오픈 카드이며 다음 플레이어는 이 오픈 카드의 무늬와 숫자, 또는 공격포인트에 따라 카드를 내야 한다.

V. 낼 수 있는 카드가 없다면 오픈되어 있는 카드를 제외한 테이블상의 나머지 카드들 중에서 임의로 한 장을 가져오도록 한다. 테이블상의 나머지 카드란, 플레이어들이 가지고 있는 카드와 오픈 카드를 제외한 카드를 말한다. 낼 수 있는 카드가 있는 경우에도 플레이어가 원한다면 한 장의 카드를 가져갈 수 있다.
처음에 배분하고 남은 카드들과 이 후 플레이어들에 의해 쌓인 카드들을 구분 없이 하나의 덱(이후 main deck)에서 관리하자. (Tips 참조)
※ **main deck**: 플레이어들이 가지고 있는 카드를 제외하고 오픈된 카드를 포함한 테이블 상의 모든 카드들의 모음. (이 과제에서 설명을 위해 main deck에 오픈 카드를 포함시켰으나, main deck은 오픈 카드를 제외하고 구현해도 무방하다.)

VI. 공격 카드

다음에 해당하는 카드를 내면 공격이 시작되고 공격 포인트가 쌓인다. 이를 최종적으로 방어하지 못한 플레이어가 오픈 카드를 제외한 main deck에서 임의로 누적된 공격 포인트만큼 카드를 가져간다.

i. **2(투)**: 다음 플레이어가 오픈 카드를 제외한 main deck에서 2장을 가져간다. 다른 무늬의 2(투), 같은 무늬의 A(에이스), Joker(조커)로 방어할 수 있다.
ii. **A(에이스)**: 다음 플레이어가 오픈 카드를 제외한 main deck에서 3장을 가져간다. 다른 무늬의 A(에이스), Joker(조커)로 방어할 수 있다.
iii. **♤A(스페이드 에이스)**: 다음 플레이어가 오픈 카드를 제외한 main deck에서 5장을 가져간다. 다른 무늬의 A(에이스), Joker(조커)로 방어할 수 있다.
iv. **Joker Black(흑백 조커)**: 다음 플레이어가 오픈 카드를 제외한 main deck에서 7장을 가져간다. Joker Color(컬러 조커)로 방어할 수 있다.
v. **Joker Color(컬러 조커)**: 다음 플레이어가 오픈 카드를 제외한 main deck에서 10장을 가져간다. Joker Black(흑백 조커)로 방어할 수 있다.

ex) 이전 플레이어가 공격카드에 해당하는 [♤ 2]를 내면 다음 플레이어는 [♤ J]등의 카드를 낼 수 없다. 대신 [♧ 2], [♥ 2], [◆ 2], [♤ 2], [♤ A], [JK B] (흑백조커), [JK C] (컬러조커) 등의 카드로 방어하거나 2장의 카드를 main deck에서 가져가야 한다.

(실행예제 Figure 3, 5 참조)

VII. 공격 카드를 방어하지 못한 플레이어가 main deck(오픈 카드 제외)에서 카드를 가져가면 다음 플레이어는 원래 규칙대로 같은 숫자, 또는 같은 문양의 카드를 낼 수 있다. 단, 흑백조커가 마지막 공격카드 이었을 경우에는 숫자와 상관없이 ♤ 또는 ♧ 무늬의 카드를, 컬러조커가 오픈되어 있을 경우에는 ♥ 또는 ◆ 무늬의 카드를 낼 수 있다. 이 규칙은 처음에 오픈한 카드가 조커일 경우에도 동일하게 적용된다. **(실행예제 Figure 4, 6 참조)**

VIII. 플레이어가 가진 카드가 20장 이상이 되면 해당 플레이어는 **파산**한다. 이 경우 파산한 플레이어의 카드는 main deck으로 가게 되고 이 플레이어를 제외한 나머지 플레이어들은 게임을 이어나간다. **(실행예제 Figure 12 참조)**

> ex) 'A', 'B', 'C'세 명이 플레이를 하고 있고 순서는 A->B->C->A->... 인 상황을 가정하자. 현재 'C'는 5장의 카드를 들고 있다. 이때, 'A'가 [JK B]를 내면 'B'는 [JK C]를 내서 방어하거나 7장을 덱에서 가져가야 한다. 만약 'B'가 [JK C]를 내서 방어한다면 'C'는 방어할 수 있는 수단이 없으므로 덱에서 (7+10)장을 가져가야 하고 이는 곧 'C'가 파산함을 의미한다. (7+10+5>=20) 'C'가 파산한 후에는 'C'가 들고 있던 모든 카드를 main deck에 반납하게 되고 원래 순서로 다음 플레이어인 'A'는 컬러조커가 오픈되어 있으므로 ♥ 또는 ◆ 무늬의 카드를 낼 수 있다.

IX. 특수 카드

i. **J(점프)**: 게임의 순서 방향(처음에는 반 시계방향)으로 한 플레이어를 건너 뛴다. 2명이서 플레이 하는 경우 한 장을 더 낼 수 있는 효과를 낸다.
(실행예제 Figure 8, 9 참조)

ii. **Q(퀸)**: 게임의 순서 방향을 바꾼다. 즉 처음 시작이 시계반대 방향이면 Q를 낸 다음에는 시계방향으로 게임의 순서가 바뀐다. 2명이서 플레이 하는 경우 아무 효과가 없다.
(실행예제 Figure 10 참조)

iii. **K(킹)**: 한 장의 카드를 더 낼 수 있다. 즉, 플레이어 'A' 가 K를 냈다면, 다시 플레이어 'A' 차례가 되는 것이다. K를 낸 후, 낼 수 있는 카드가 없으면 카드를 가져가야 한다. 한 플레이어에게 K가 4장이 있는 경우 최대 5장의 카드를 연속으로 낼 수 있다.
(실행예제 Figure 11 참조)

iv. **7**: 카드의 무늬를 바꿀 수 있다. 현재 무늬 그대로 유지하는 것도 가능하다.
(실행예제 Figure 7 참조)

X. 카드가 한 장 남았을 때 다른 플레이어보다 "원카드"를 먼저 외치지 못하면 카드 한 장을 가져가야 하는 규칙은 **제외**한다. 단, 실제 구현에 있어서 카드가 한 장이 남게 되면 이를 알리는 간단한 문구를 출력해준다. **(실행예제 Figure 16 참조)**

XI. 만약 (공격 등으로)플레이어가 가져가야 하는 카드의 수 보다 main deck에 남아있는 카드의 수가 적은 경우에는 오픈 카드를 제외한 main deck의 카드 전부를 해당 플레이어가 가져가도록 한다. 오픈 카드를 제외하고 main deck의 카드가 없는 경우에는 카드분배 없이 게임을 진행한다. (카드를 분배하지 않아도 반드시 게임을 진행할 수 있는 플레이어가 존재한다.) **(실행예제 Figure 13, 14 참조)**

XII. 게임종료

게임은 모든 플레이어가 파산하고 단 하나의 플레이어가 남은 경우, 또는 어떤 플레이어가 가장 먼저 모든 카드를 내려놓는 경우에 해당플레이어의 승리로 종료된다.
(실행예제 Figure 15, 16 참조)

(요구사항)

- 프로그램 구현은 **구조체와 linked list**를 사용하여야 하며 제약사항은 다음과 같다.
  - CARD: 아래와 같은 자기참조 구조체로 구현한다.

```
typedef struct card{
    char* cardPattern;
    char cardNumber;
    struct card* nextCard;
}CARD;
```

- cardPattern은 특수문자를 포함한다 : "♧", "♥", "◆", "♤", "JK" (JK는 조커를 의미)

- cardNumber는 int 또는 char 중 어떤 변수형으로 사용해도 무방하다. 단, 출력시에 'A', '2', ..., '9', '0', 'J', 'Q', 'K', 'B', 'C' 로 출력되도록 한다. ('0'은 10을, B는 흑백(조커)을, C는 컬러(조커)를 의미)

  - DECK: 아래와 같은 Linked list로 구현한다.

```
typedef struct deck{
    CARD* card;
    int count;
}DECK;
```

- count는 deck에 포함된 카드의 개수를 의미한다. 덱에서 카드를 삽입하고 삭제할 때 count 관리에 유의한다.

  - PLAYER: 아래와 같은 양방향 linked list로 구현한다.

```
typedef struct player{
    char name;
    DECK* deck;
    struct player* right;
    struct player* left;
}PLAYER;
```

- name은 사용자 입력을 받아서 설정하는 것이 아닌 자동으로 설정 되도록 한다. (2명일 경우에는 'A', 'B', 4명일 경우 'A', 'B', 'C', 'D')

- deck은 각 사용자가 가지고 있는 deck의 포인터이다.

- right와 left는 각각 오른쪽, 왼쪽의 플레이어를 가리키는 포인터로 항상 NULL값이 되지 않도록 관리한다. (어느 누가 파산했을 경우에도)

필요하다면 위 구조체들 내에 변수들을 적절히 추가 또는 수정하여 활용하도록 한다.

- 플레이어가 들고 있는 카드나 테이블에 쌓여있는 카드들은 정렬될 필요가 없다.
- 메모리 관리를 효율적으로 한다. 즉, 불필요하거나 중복되는 할당을 하지 않도록 한다. (본 과제에서는 프로그램 초기에 필요한 메모리를 할당하고 나면 그 이후에는 추가적인 메모리할당이 불필요하다. 카드나 덱이 추가적으로 생성될 일이 없기 때문이다. **메모리 해제**도 플레이어가 파산할 때와 프로그램이 종료되는 시점에만 하는 것으로 충분하다.)
- **반드시 linked list로 구현해야 하는 부분: main deck, 각 플레이어의 덱, 게임에 참여하고 있는 플레이어들의 리스트**

➢ 구현상의 세부적인 이슈에 대한 가이드라인을 정리하자면 다음과 같다

- 모든 덱에 포함된 카드의 수는 어떠한 경우에도 54 장이 되어야 한다.
- 기본적으로 덱은 main deck 과 각 플레이어들이 소지하고 있는 덱이 있다.
- 처음 카드를 배분하고 섞는 방법 및 순서는 구현하는 사람의 자율이다.
- 처음에 플레이어들에게 배분하고 남은 카드와 이 후 플레이어들이 내는 카드 구별 없이 **하나의 덱에서 다룬다.**
- 오픈 카드는 main deck 에서 하나의 카드를 오픈 카드로 정하여 관리할 수도 있고 또는 별도로 관리해도 무방하다. 본 예제에서는 오픈 카드를 main deck 의 가장 첫 카드로 지정하여 관리하고 있다. (삽입이 앞에서 이루어지므로)
- main deck 이나 플레이어의 덱에 카드가 삽입될 때, 앞에서 삽입해도 되고 뒤에서부터 삽입해도 된다. 이는 구현하는 사람의 자율이다.
- main deck 에서 카드를 삭제하는 경우에도 앞, 뒤 또는 임의로 중간에서 삭제해도 된다. 단, 이 경우에는 **오픈 카드는 유지되어야 하고 '임의로' '필요한 개수'만큼의 카드를 가져오는 것이 중요**하다.
- Debugging 용 출력(아래 출력 예제에서 주황색 박스 안에 출력된 내용)에서는 **현재 프로그램 내에 존재하는 모든 덱과 해당 덱에 들어있는 카드의 개수 및 카드들이 출력**되어야 한다. 만약 오픈 카드를 별도로 관리한다면 해당 오픈 카드도 출력해야 한다.

➢ **출력 형태**는 본 예제와 최대한 비슷하게 한다. 이 때 **반드시 출력되어야 하는 요소는 아래와 같다.** (출력 내용 이외에 빈칸 및 정렬상태는 다소 달라도 괜찮다)

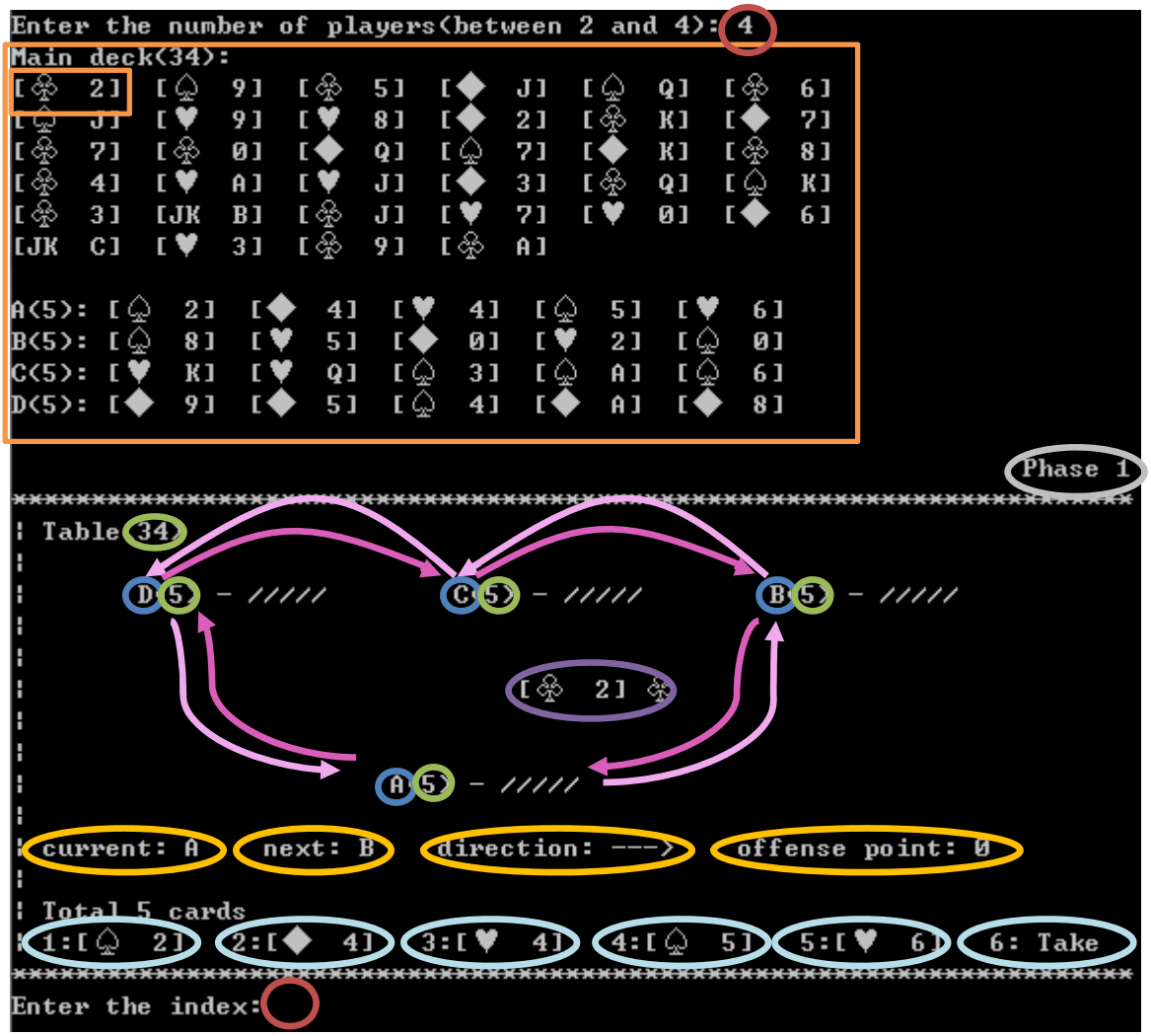


Figure 1

- 처음 프로그램이 실행되면 플레이어의 숫자를 2~4 사이의 정수 값으로 입력한다. (프로그램 내 모든 입력에 대해 잘못된 입력은 고려하지 않는다. 즉, 예외처리는 안 해도 됨)
  : 사용자 입력
- 매 턴마다 위 형태의 테이블과 카드정보를 출력하고 카드를 내기 위한 인덱스 값을 입력받기 위해 대기한다. 여기에는 다음과 같은 정보가 있어야 한다.

  ① **테이블 출력 전**, main deck의 모든 카드와 카드 개수(현재 34장) 및 모든 플레이어들과 플레이어들이 가지고 있는 카드의 개수 및 카드를 출력한다. 이 때, 플레이어의 출력 순서는 현재 플레이어(A)가 가장 먼저 출력되고 그 이후 플레이 할 플레이어들이 이 순서대로 출력된다. 이 예제에서는 게임을 처음 시작하는 상황이므로 플레이어A가 현재 플레이어인 상태이고 게임 방향은 시계반대방향이므로 플레이 순서는 A->B->C->D 이다. 따라서 A, B, C, D 순으로 플레이어가 가진 카드를 출력한다. 현재 main deck의 첫 번째 카드 **[♧ 2]**가 오픈 카드로 지정되어 있음을 알 수 있다.

이 출력은 디버그 및 채점을 용이하게 하기 위한 것으로 반드시 게임룰에 맞게 출력이 되어야 한다. (카드의 총 합이 54장이 되지 않거나 플레이어의 순서, 및 소지하고 있는 카드가 잘못 된 경우 등 게임 룰에 맞지 않게 출력된 경우 비록 내부 알고리즘이 제대로 작동된다고 하더라도 감점이 있음)

② 테이블 위쪽에는 다른 플레이어들의 이름을, 오픈 되어있는 카드 아래쪽에는 현재 플레이어의 이름을 나타낸다. 이 때 플레이어의 이름은 **시계 반대방향 순서대로 출력하도록 한다.**

※ 예제에서 진한 분홍색 화살표는 각 플레이어들의 왼쪽 플레이어를, 연한 분홍색 화살표는 오른쪽 플레이어를 나타낸다. 실제 출력에서는 화살표가 없지만 플레이어들의 이름을 순서대로 출력하여 직관적으로 플레이어들간의 참조상태를 알 수 있도록 배치하도록 한다.

③ 테이블의 카드 수(오픈 카드 + 배분하고 남은 카드 + 플레이어가 낸 카드)와 각 플레이어가 현재 가지고 있는 카드 수를 각각 표시한다. 전체 카드는 54 장이므로 이들 **카드수의 합은 항상 54장**이어야 한다. (현재 카드를 5장씩 배분한 직후 이므로 테이블에는 34장(오픈카드 포함), 각 플레이어당 5장씩 총 54장의 카드가 있다.)

④ 테이블의 중앙에는 현재 오픈 되어있는 카드와 그 카드의 오른쪽에 다음에 내야 할 패턴을 표시한다.
(숫자 10 은 0으로 나타내도 무방하다. 또한 흑백조커의 숫자 칸에는 B를, 컬러조커의 숫자 칸에는 C를 표시한다)

⑤ 테이블의 아래쪽에는 현재 플레이어, 다음 플레이어의 이름과 방향, 공격 포인트를 표시한다. (위 예제에서 현재 플레이어는 'A', 다음 플레이어는 'B'이고 방향은 오른쪽, 공격 포인트는 0임을 나타낸다)

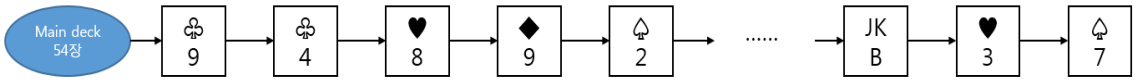
⑥ 가장 아래쪽에 현재 플레이어가 가지고 있는 카드를 출력한다. 이 때 각 카드 왼쪽에는 인덱스를 출력하며 마지막+1 번째에는 더 이상 낼 카드가 없거나 플레이어가 카드를 얻기를 원할 때 입력하기 위한 인덱스를 제공한다.

⑦ 마지막으로 현재 진행중인 게임상황을 알 수 있도록 몇 번째 턴인지 나타내는 숫자를 출력한다. (예제에서 오른쪽 상단, Phase1)

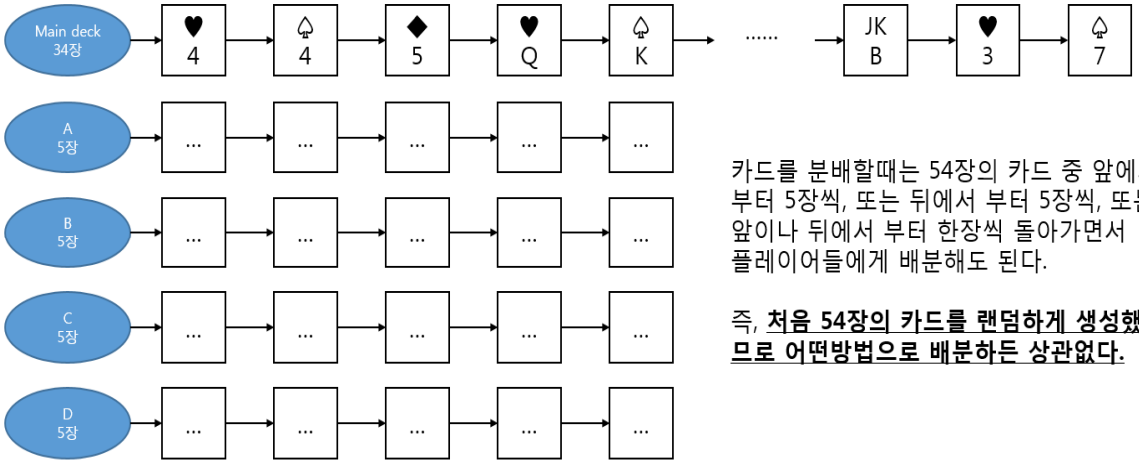
Tips

➢ **플레이어 숫자를 입력하고 처음 카드를 배분할 때 카드 및 덱의 형태 예시는 다음과 같다.**

1. 54장의 카드를 랜덤하게 생성



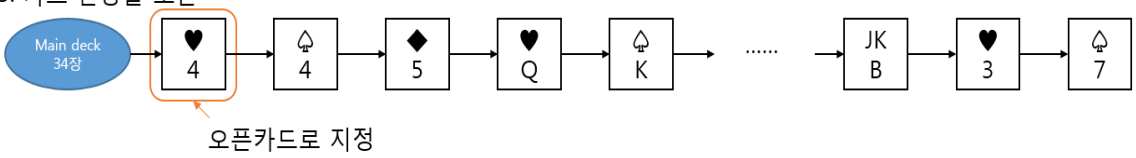
2. 플레이어들에게 카드 5장씩 배분



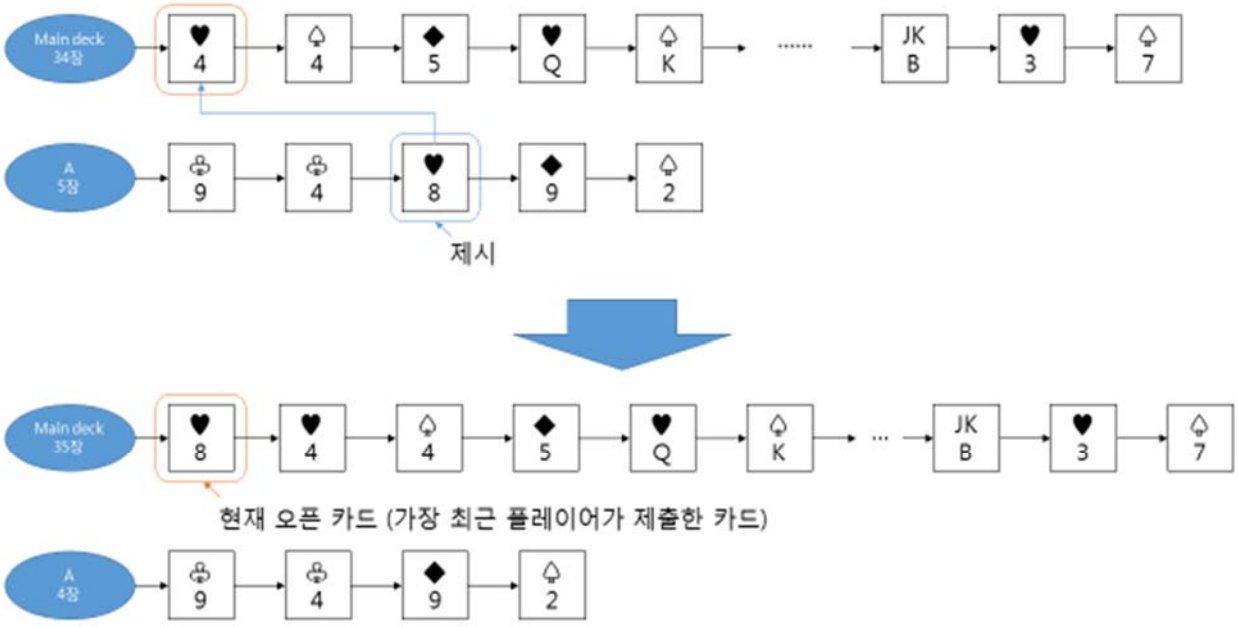
카드를 분배할때는 54장의 카드 중 앞에서 부터 5장씩, 또는 뒤에서 부터 5장씩, 또는 앞이나 뒤에서 부터 한장씩 돌아가면서 플레이어들에게 배분해도 된다.

즉, **처음 54장의 카드를 랜덤하게 생성했으므로 어떤방법으로 배분하든 상관없다.**

3. 카드 한장을 오픈

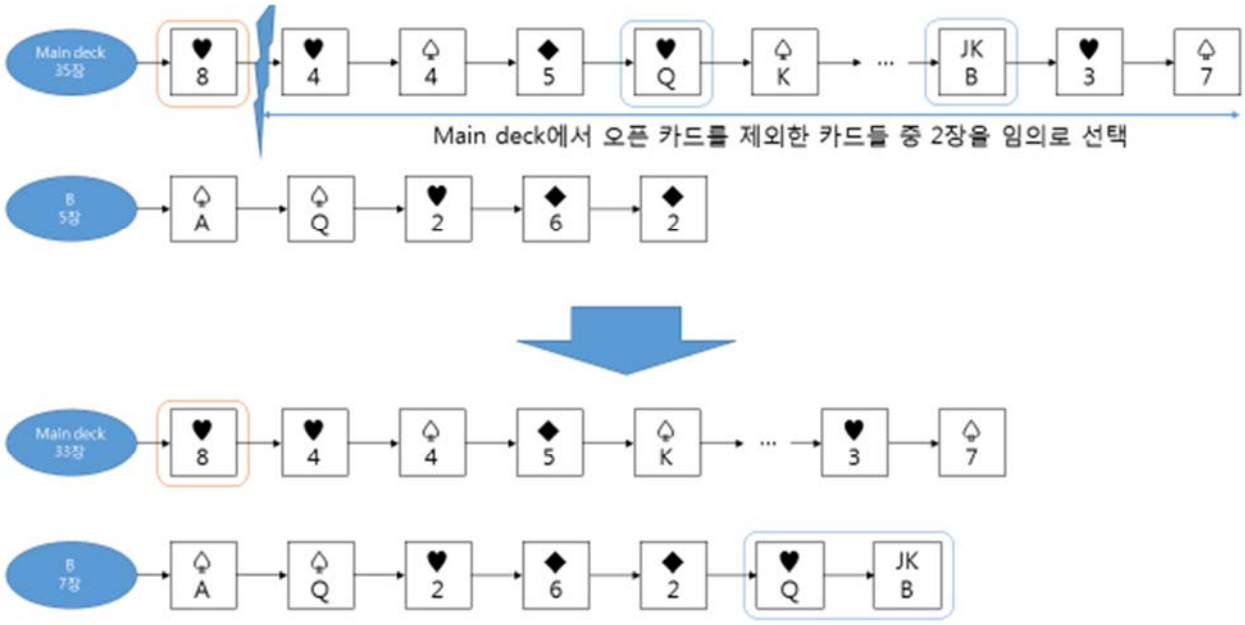


➢ **플레이어 'A' 가 규칙에 맞는 카드를 낸 경우의 예시는 아래와 같다.**

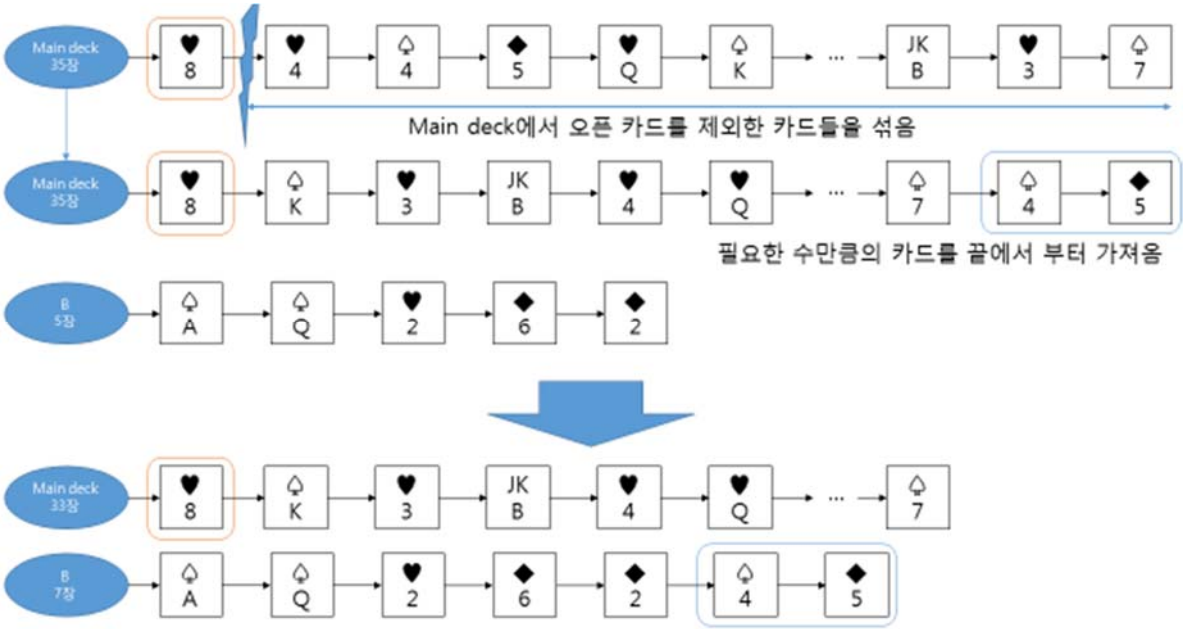


➢ **공격을 받고 수비에 실패한 플레이어 'B'가 2장의 카드를 main deck으로부터 카드를 가져가는 경우**

예시 1)



예시 2)



덱으로부터 카드를 가져올 때, 예시 1)과 예시 2) 중 어떤 방법으로 구현해도 상관없다. 또는 다른 방법을 사용해도 무방하다. 중요한 것은 오픈 카드는 항상 유지되어야 하며 이를 제외하고 main deck으로부터 필요한 수만큼의 카드를 **랜덤하게 선택**해서 가져와야 한다는 것이다.
(플레이어가 가지고 있는 카드는 정렬하지 않아도 된다.)

(실행예제)

아래 실행예제에서는 지면 관계상 테이블만 출력하고 debugging을 위한 main deck과 플레이어들이 소지한 카드들은 출력하지 않고 있다.

하지만 **실제 구현에서는 모든 턴마다 디버그를 위한 출력을 반드시 출력하도록 한다.**

1. 기본적인 패턴 및 숫자 매칭

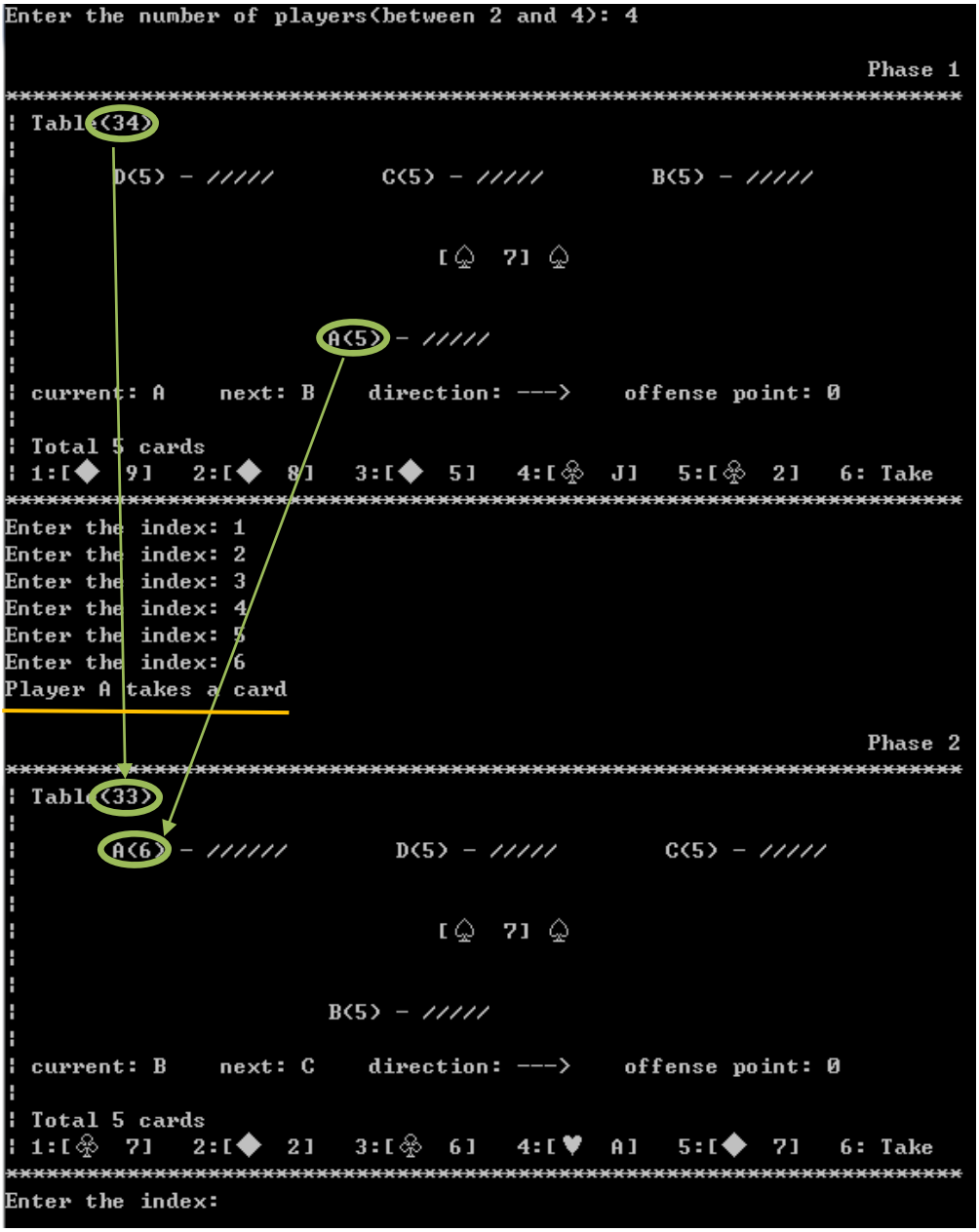


Figure 2

- Phase1 에서 Player A는 적절한 카드가 없으므로 한 장을 테이블에서 가져간다. 따라서 Phase2 에서 A가 가진 카드의 수는 6개가 되며 테이블의 카드 수는 33장이 된다. 카드의 개수나 플레이어들의 배치는 현재 상황에 맞도록 출력되어야 한다.
- 잘못된 카드(같은 무늬 또는 같은 숫자나 같은 영문이 아닌 카드, 공격중인데 방어할 수 없는 카드)를 제시하는 경우, **규칙에 맞는 카드를 내거나 카드를 가져갈 때까지 입력을 다시 받도록 한다. 잘못된 카드를 제시해도 패널티는 없다.**

2. 공격

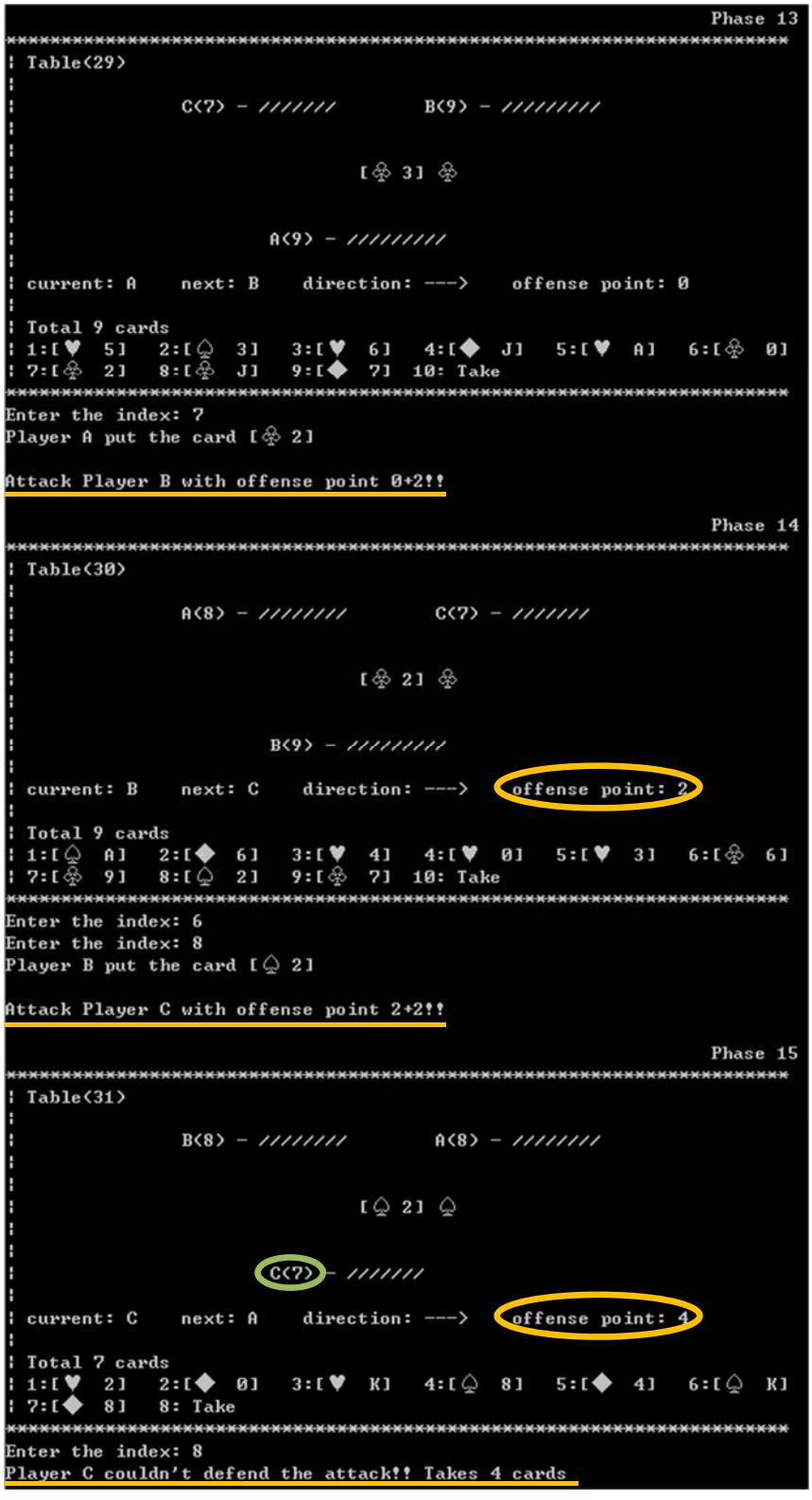


Figure 3

- A가 [♧ 2]로 공격을 시작하면 (Phase13) B는 일반적인 매칭 카드(6:[♧ 6])를 낼 수 없다. 대신 같거나 더 높은 레벨의 공격카드 (8 :[♤ 2])로 방어할 수 있으며(Phase14), C가 이를 방어하지 않고 카드를 받고자 하면 4장(=2+2)의 패널티를 받게 된다.(Phase15) 물론 위 예제에서는 C역시 1:[♥ 2] 카드로 방어하고 패널티를 피할 수 있다.

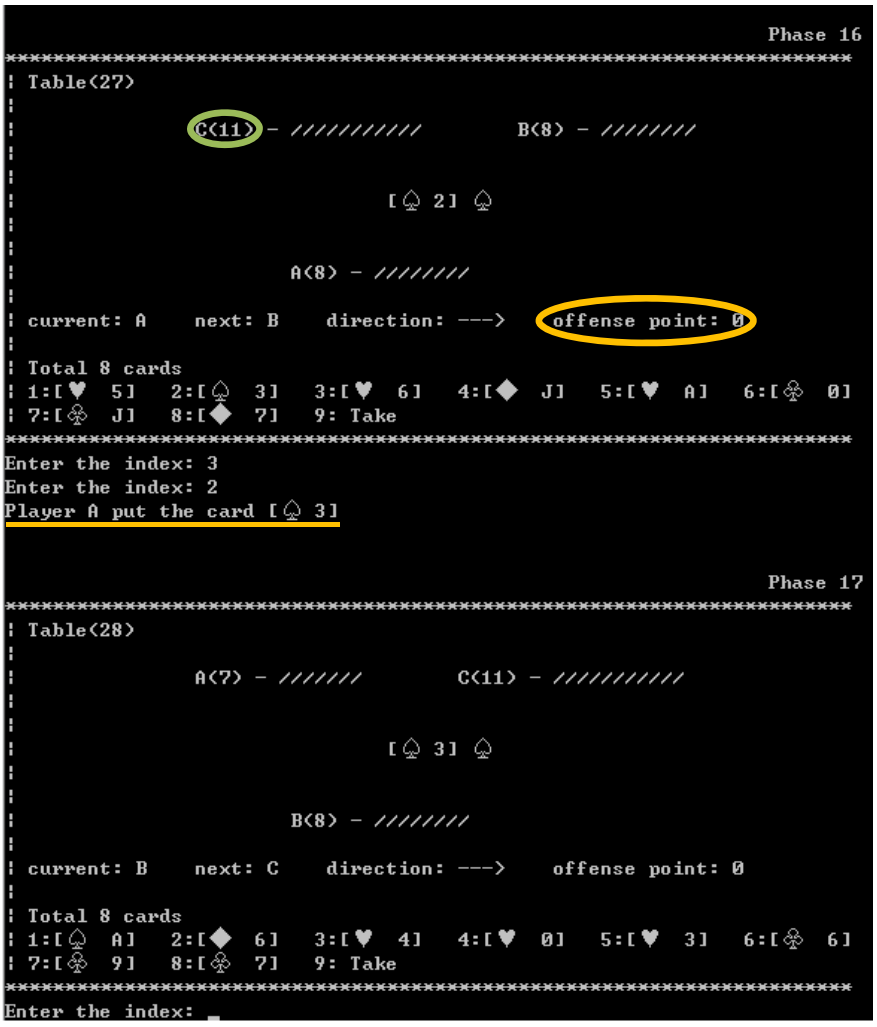


Figure 4

- 한편, 공격이 끝나고 누군가 패널티를 받게 되었으므로 다음 플레이어인 플레이어 A는 일반적인 매칭 규칙대로 2:[♤ 3]을 낼 수 있다(Phase16). 이 때 C는 원래 7장에서 4장의 패널티를 더 얻게 되어 총 11장의 카드를 보유하고 있음을 알 수 있다.

Figure 5

- 조커의 경우에도 마찬가지 규칙이 적용된다. Figure 5 에서 플레이어 B가 조커 4:[JK B]를

내고(Phase5), Figure 6에서 C는 이를 방어하지 못해 7장의 카드를 받은 상황을 보면(Phase6) 공격을 방어했거나 아직 패널티를 받지 않은 상황에서는 offense point가 7이고 조커를 제외한 다른 카드를 낼 수 없다(Phase6). 하지만 C가 패널티를 받은 다음에는 A는 ♤ 또는 ♧ 무늬의 일반카드를 낼 수 있다(Phase7). (이 때 공격 포인트는 0이다.)

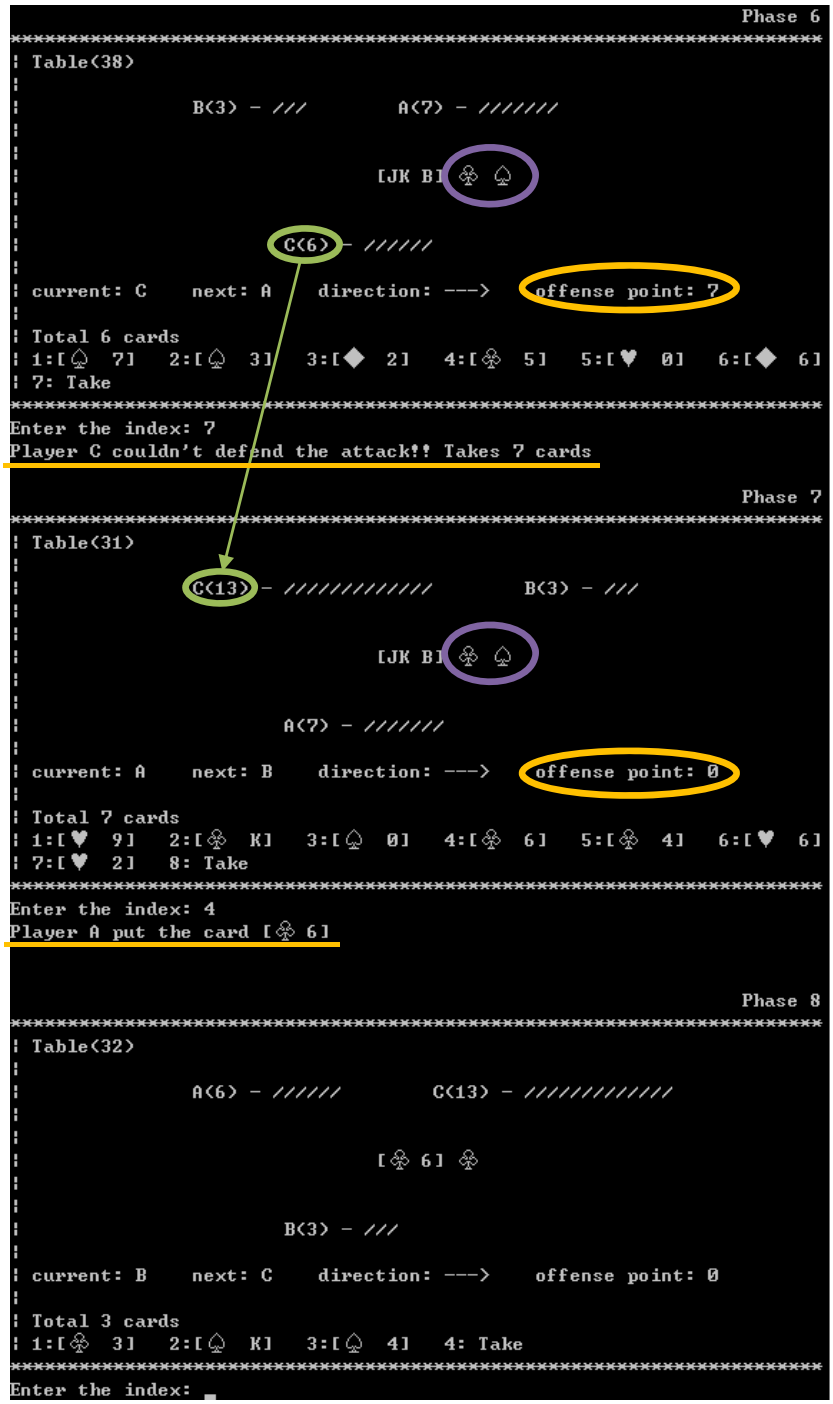


Figure 6

3. 특수카드(7)

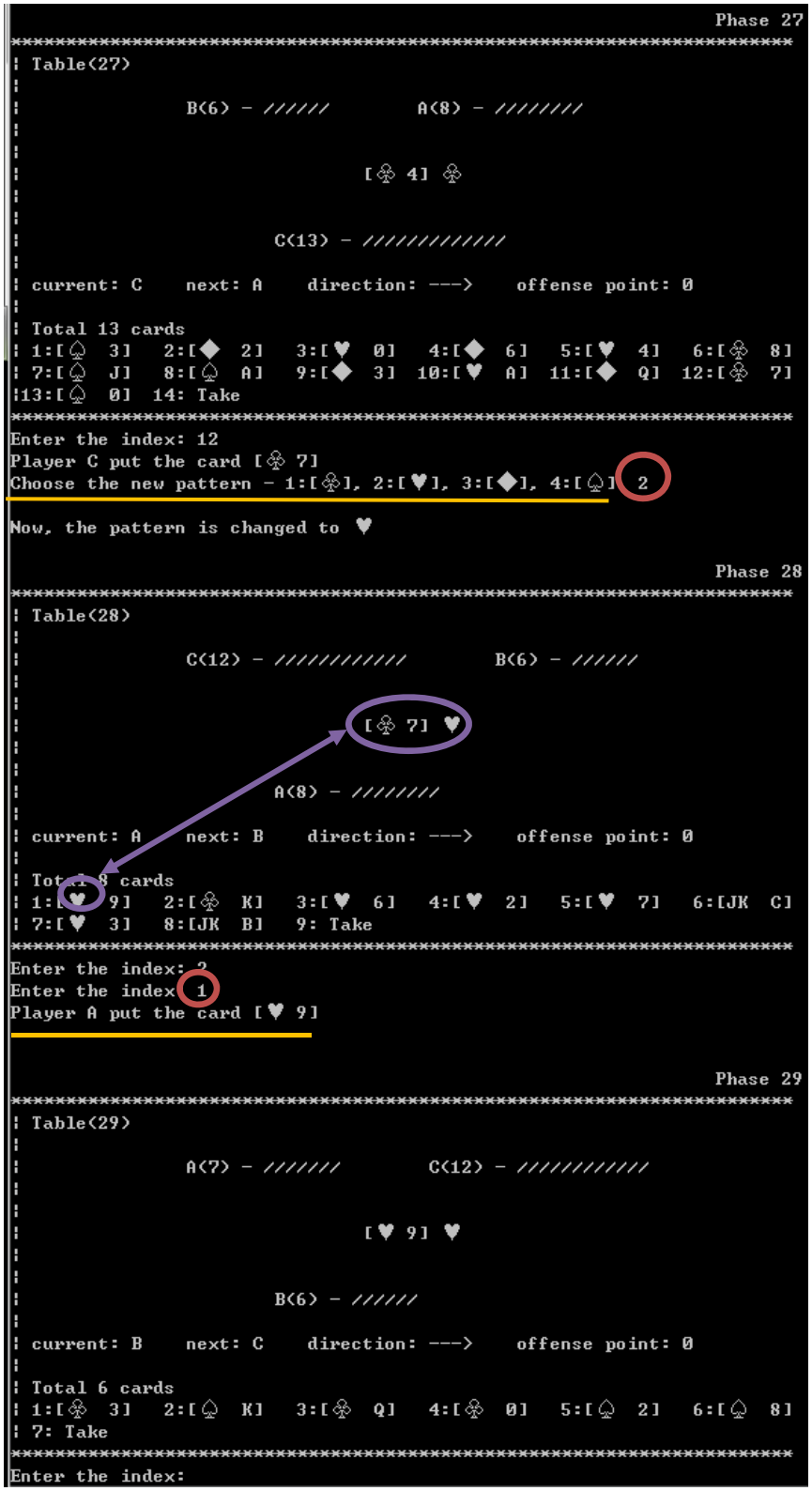


Figure 7

- 플레이어가 숫자 7 카드를 내면 새로운 패턴을 지정하기 위해 사용자입력을 기다린다(Phase27). 새로운 패턴을 지정하고 나면 플레이어 A는 기존의 ♧와 매치되는 2:[♧ K]를 낼 수 없고 바뀐 패턴(♥)에 맞는 1:[♥ 9]를 낼 수 있다.(Phase28)

4. 특수카드 J(점프)

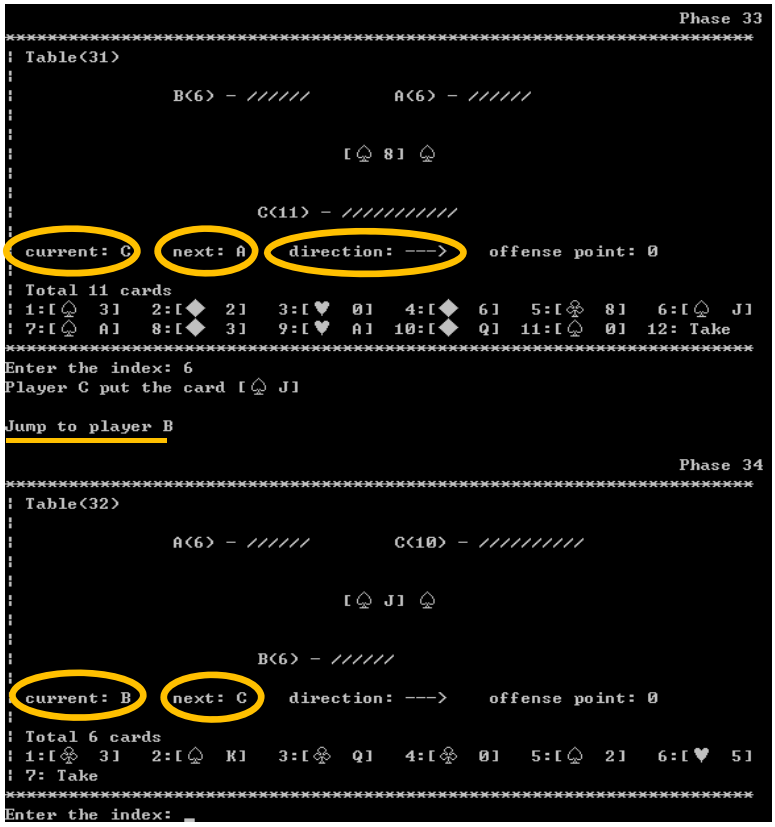


Figure 8

← 3인이상 플레이:
점프하면 현재 방향으로(오른쪽) 한 플레이어를 건너뛰게 된다(Phase33)

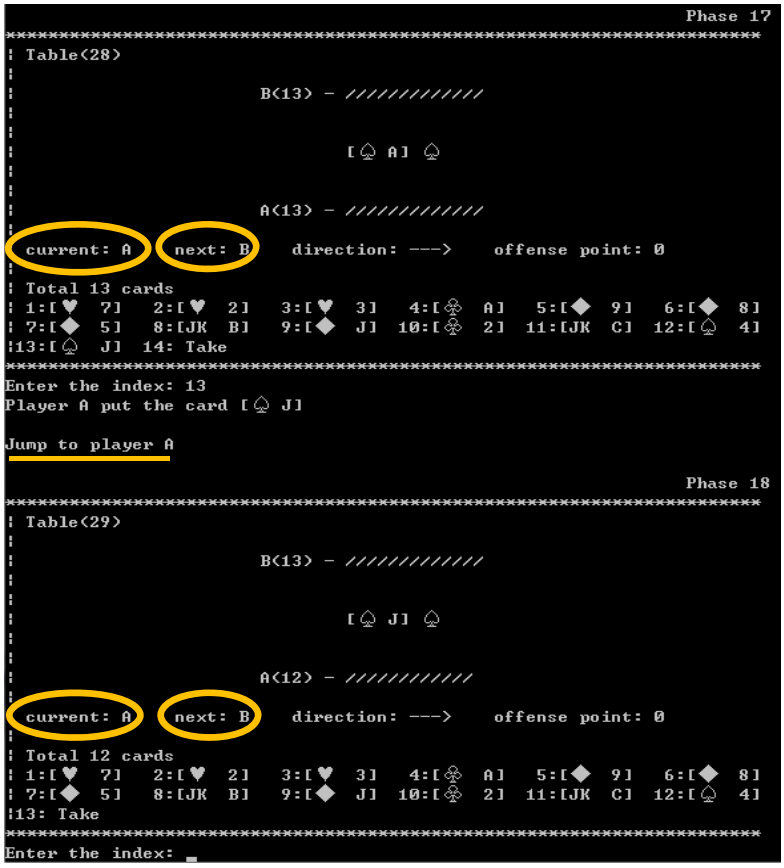


Figure 9

← 2인 플레이:
점프는 한 번 더 턴이 돌아오는 효과를 갖는다(Phase17)

5. 특수카드 Q(퀸)

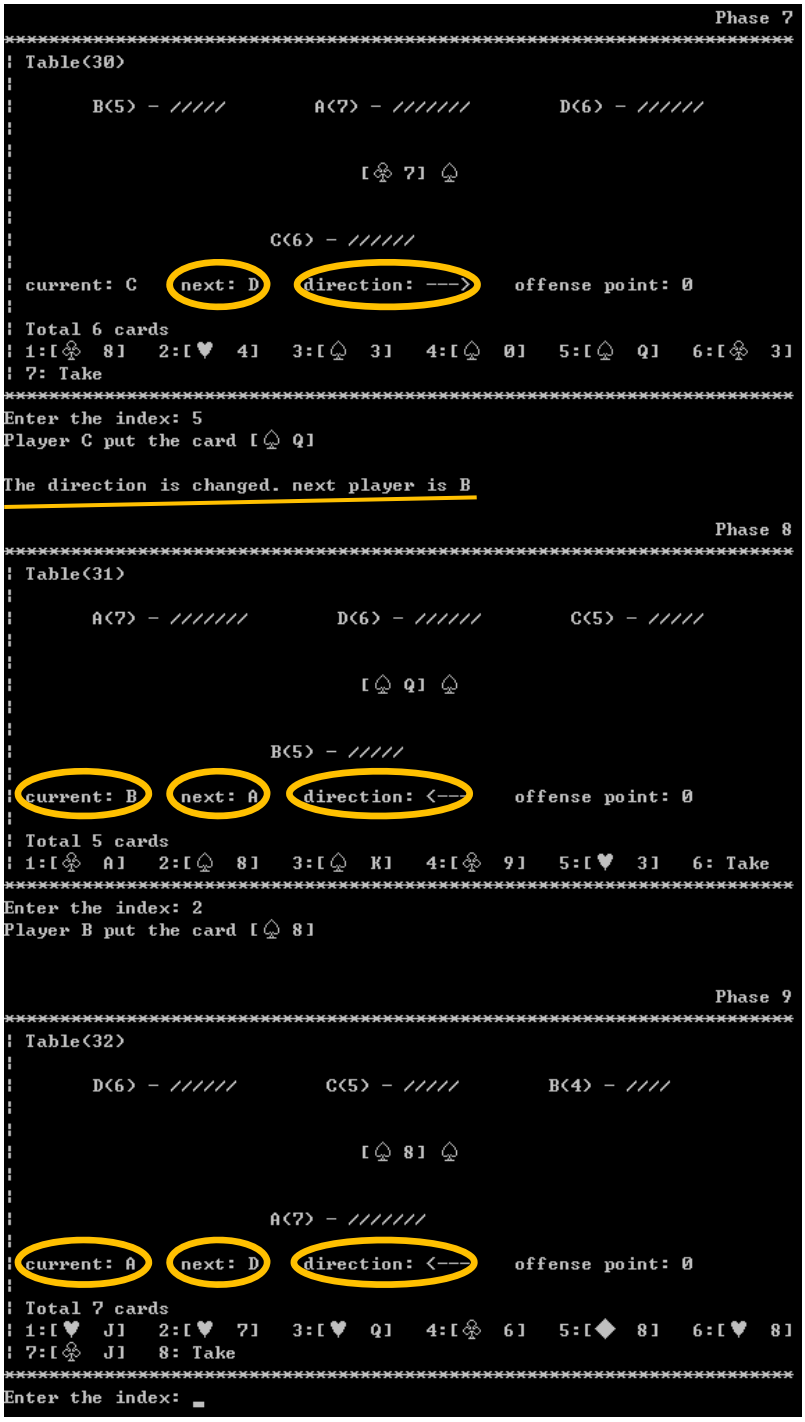


Figure 10

- 현재 방향이 오른쪽일 때 플레이어 C가 Q를 내면 방향이 자동으로 왼쪽으로 바뀌고(Phase7) 다음 플레이어는 D가 아닌 B가 된다. 현재 플레이어가 B인 상황에서(Phase8) 다음 플레이어(next)는 바뀐 방향(왼쪽)을 고려하여 A로 나타나게 된다. 이 후 또 다시 Q를 내면 방향은 다시 오른쪽으로 바뀌게 되고 바뀐 방향에 따라 다음플레이어가 결정된다.
  (이 때 Debugging용 출력에서는 B->A->D->C 순으로 플레이어들이 출력되어야 한다.(Phase8))

6. 특수카드 K(킹)

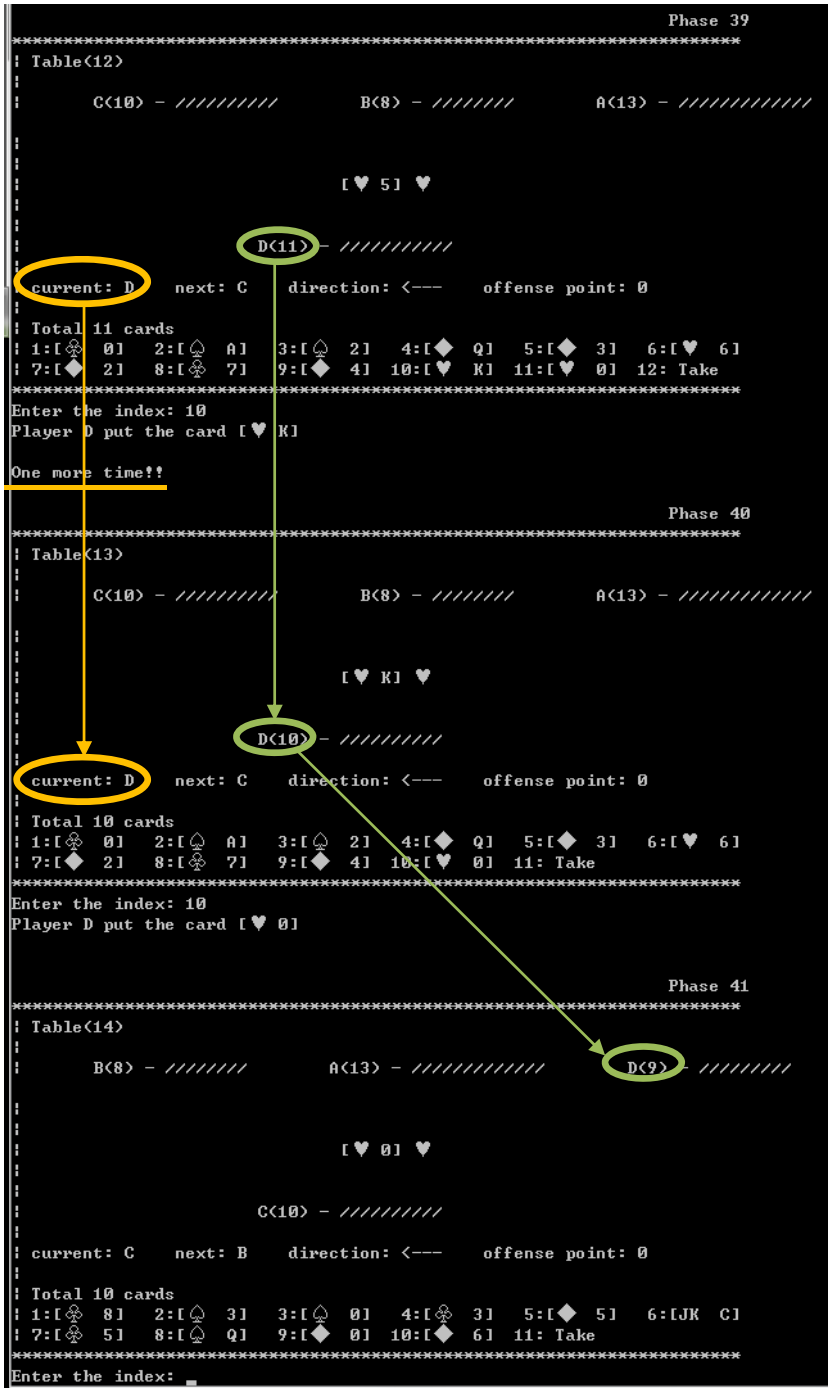


Figure 11

- 플레이어 D가 10:[♥ K]를 내면 한번 더 턴이 돌아오게 되고 한번 더 패를 낼 수 있는 기회를 가진다(Phase39, 40).

7. 파산(Bankrupt)

Figure 12

- 현재 상황에서 플레이어 C가 [JK C]를 내게 되면 다음 플레이어인 B는 공격포인트 10을 받게 된다(Phase 49). B는 방어할 수 있는 카드가 없고 10장의 카드를 소지하고 있으므로 이 공격을 그대로 받게 되어 파산하게 된다(10+10 >= 20)(Phase50). 플레이어 B가 파산한 뒤에는 게임에서 빠지게 되고, B가 가지고 있던 모든 카드는 테이블(main deck)로 가게 된다(7->17)(Phase51). 이 후에는 원래 순서대로(왼쪽) 다음 플레이어인 A가 나머지 두 명과 게임을 계속 이어나가게 된다(Phase51). 또한 플레이어가 한 명 줄어든 것 외에

다른 조건들(게임 순서 방향, 각 플레이어의 카드 개수)은 이전과 동일하게 적용되고 공격으로 인해 파산하는 경우 offense point는 0이 된다(Phase 51). (Debugging용 출력에서는 A->D->C 순서로 3명의 플레이어만 출력되어야 한다)

8. 테이블 카드가 부족한 경우

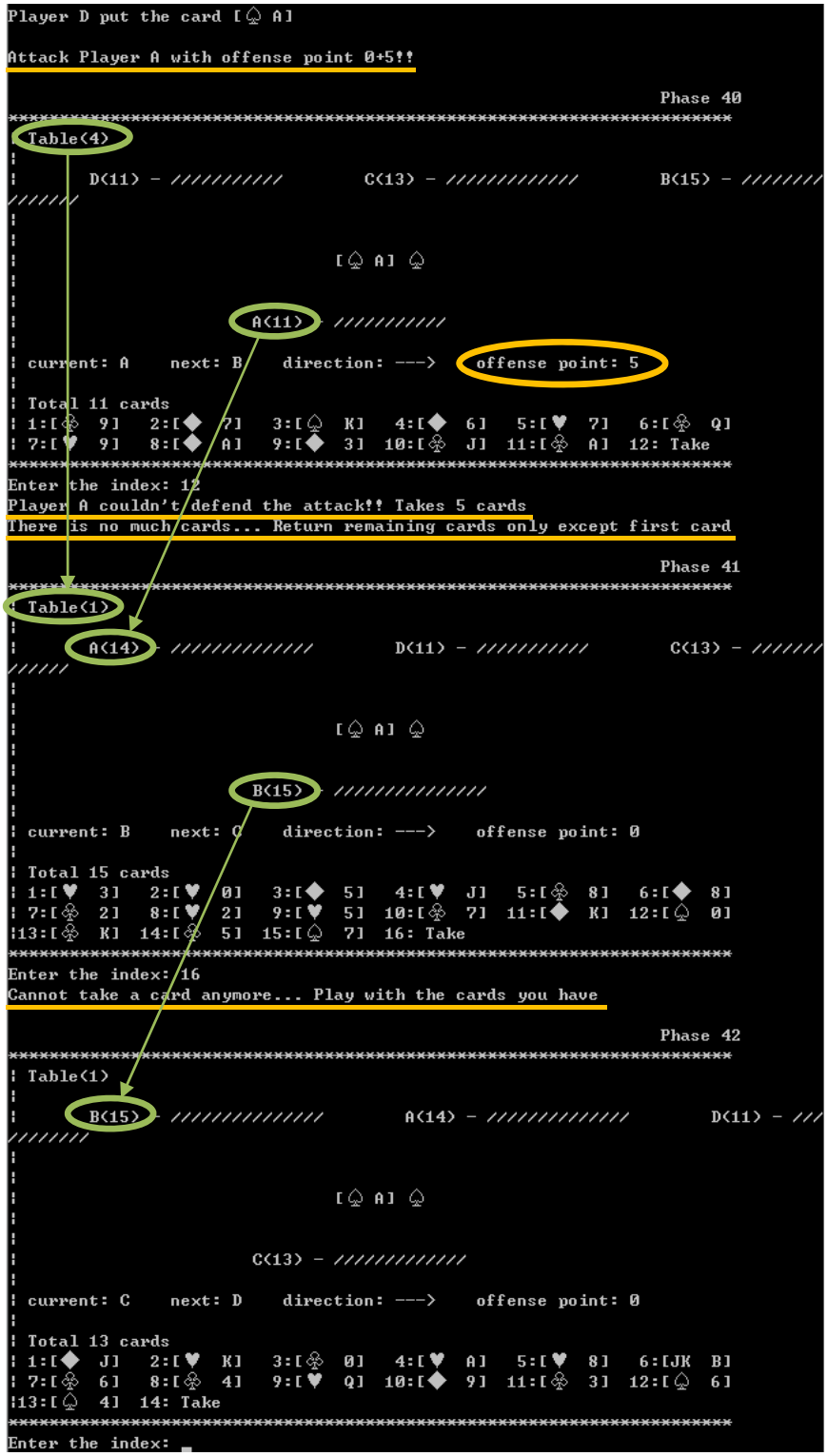


Figure 13

- 이전 Phase에서 플레이어 D가 스페이드A카드로 A에게 공격을 가했을 때, 이를 방어하지

못하더라도 덱에 충분한 카드(5장)이 없으면 오픈카드를 제외한 나머지 카드 수만큼만 패널티로 받게 된다(Phase40). 따라서 테이블에는 한 장의 오픈카드만 남게되고 A는 5장이 아닌, 3장만 가져가게 된다(Phase41). 만약 테이블에서 가져갈 수 있는 카드가 없다면(오픈카드만 있는 경우) 패널티 없이 게임을 진행한다(Phase42).

단, 이 모든 경우는 현재 가진 카드 + 패널티 < 20 인 경우에 해당한다.
(Phase40: 11+5 < 20, Phase41: 15+1 < 20)

Figure 14

- 패널티를 온전히 받았을 때 파산하는 상황이라면 받을 수 있는 카드가 적거나 없더라도 그 즉시 파산하게 된다. 테이블에 3장의 카드만 있어서 2장의 카드만 패널티를 받을 수 있더라도 현재 A가 가진 14장의 카드에 17장의 패널티를 받게 되면 파산되는 상황이므로 A는 파산하게 된다(Phase44). (14 + 17 > 20)

9. 게임 종료

다음 2 가지의 경우에 해당하면 게임이 종료된다.

(1) 아래처럼 플레이어 1명을 제외한 다른 모든 플레이어가 파산한 경우, 남은 플레이어의 승리로 종료된다.

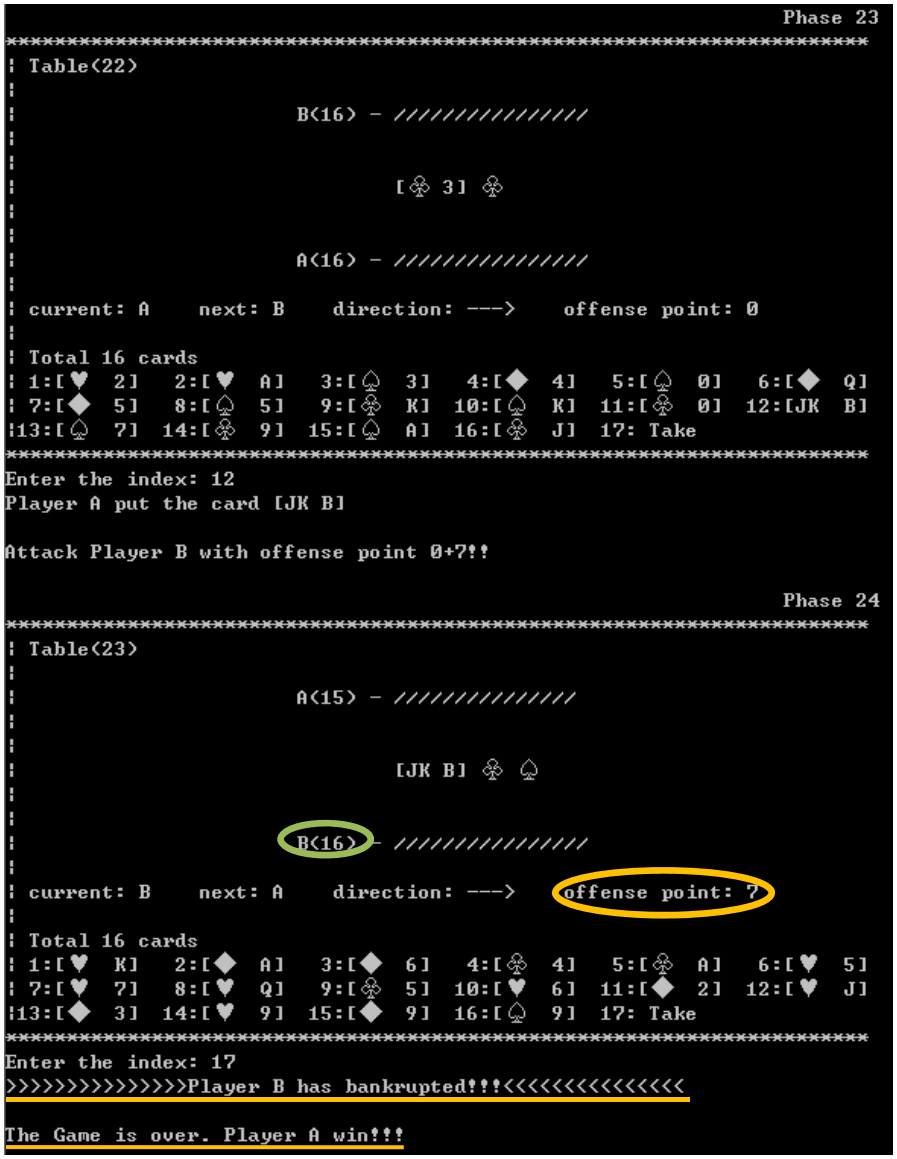
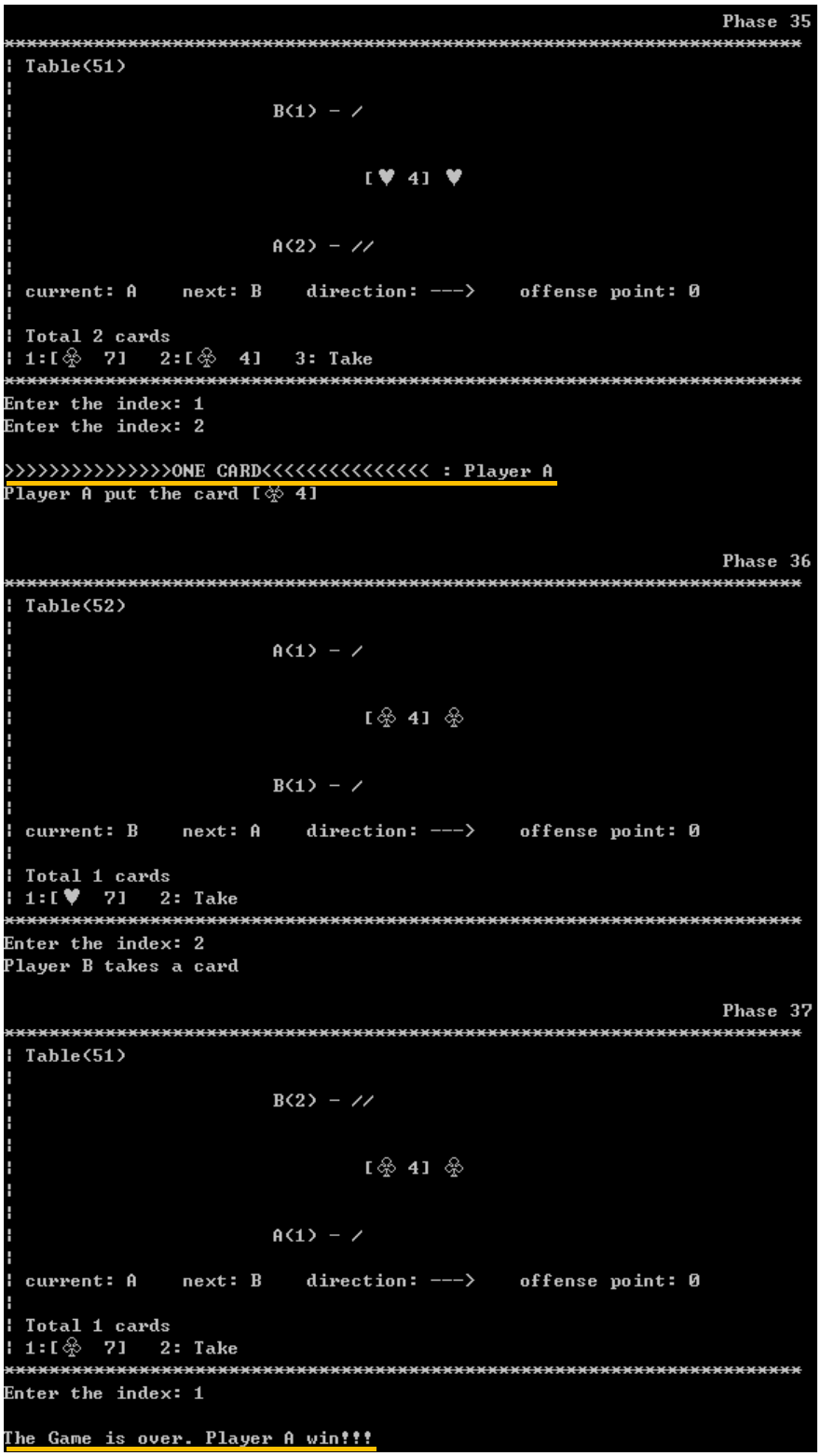


Figure 15

(2) 아래처럼 먼저 모든 카드를 내려 놓는 플레이어가 있다면, 그 플레이어가 승리하고 게임이 종료된다.



← 원카드 메시지 출력:
플레이어 A가 카드를 내고 단 1개의 카드만 가지게 되는 경우에 "원카드" 메시지를 간단히 출력한다.

Figure 16