

# CSED101. Programming & Problem solving

## Spring, 2017

### Programming Assignment #5 (70 points)

김민철(rucatia@postech.ac.kr)

■ **Due:** 2017.06.02 23:59

■ **Development Environment:** Windows Microsoft Visual Studio 2015

#### ■ 제출물

- **C Code files (\*.c and \*.h)**
  - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
- **보고서 파일** (.doc(x), .hwp or .pdf) 예) assn5.doc(x) 또는 assn5.hwp 또는 assn5.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.

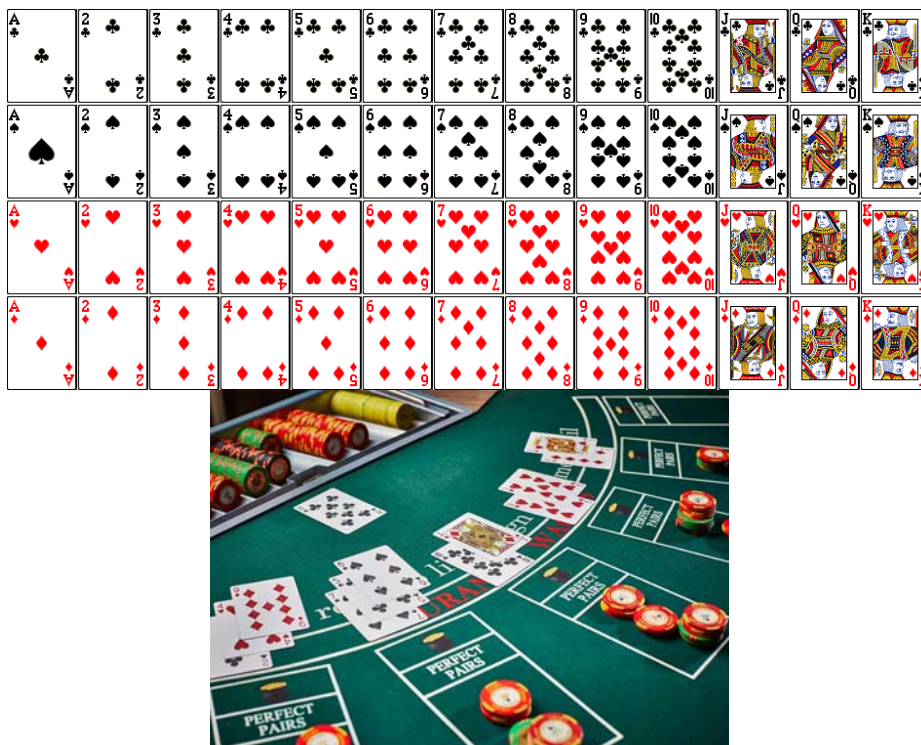
#### ■ 주의사항

- 과제에 대한 질문은 수강생들과 공유하기 위하여 LMS 질의응답 게시판을 이용한다. 이외 사항은 Office Hour를 이용하며, 부득이한 경우 E-Mail을 이용할 수 있지만 권장되지는 않으며, 질문 여부에 따라 LMS 질의응답 게시판에 올라올 수 있음을 유의한다.
- 각 문제에 해당하는 요구사항을 반드시 지켜야 할 것.
- 컴파일 & 실행이 안되면 무조건 0점 처리한다.
- 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 없다.
- 사용자 정의 함수를 적절히 선언하고 사용하여 작성하는 것을 **강력히** 추천한다.
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없다.

## ★ 블랙잭 게임 (Blackjack) ★

### (개요)

블랙잭(Blackjack) 게임은 전 세계의 어느 카지노를 가도 즐길 수 있는 가장 보편적인 플레이 카드 게임이다. 유명한 게임인 만큼 다양한 룰이 존재하며, 기본적으로는 21을 넘지 않는 숫자 내에서 딜러와 숫자의 합을 겨루어 숫자가 높으면 이기는 게임이다. 게임하는 방식 및 게임을 체험해보고 싶다면 블랙잭 게임 사이트(<http://www.247blackjack.com/>)에서 체험하는 것을 권장한다. 거의 대부분의 룰이 적용된 상태에서 플레이가 가능하다.



이번 Assignment에서는 블랙잭에서 가장 기본적인 룰인 히트(Hit), 스테이(Stay), 블랙잭(Blackjack), 버스트(Bust) 만을 적용하며, 단순한 블랙잭 게임을 구현하는 것을 목표로 한다.

### (목적)

Linked List의 기본과 Linked List가 동작하는 과정을 익히고, 학기 중에 배운 것을 토대로 종합적인 문제 해결 능력 및 약간 큰 프로그램에 대해서 효율적인 코딩을 익힌다.

### (주의사항)

- 제출해야 할 파일은 헤더 파일과 소스 파일을 포함해 총 4개로, 여러 개의 파일로 분할해서 작성한다. 확장자를 포함한 파일명과 각 파일에 들어가야 할 부분은 아래의 설명과 같다.
  - **"asn5.h"** – 구조체 타입 정의 및 함수 선언 부분
  - **"asn5.c"** – main 함수

- **"process.c"** – 아래의 구현 부분에 있는 1~6(베팅, 카드 섞기, 카드 분배, 플레이어 단계 수행, 딜러 단계 수행, 승패 판정)을 각각 수행하는 6개의 함수 정의
- **"functions.c"** – 기타 사용되는 함수 및 출력 함수의 정의
- 기타 사용되는 함수가 없다면 **"functions.c"**은 제출하지 않아도 된다. 하지만 "process.c"의 6개의 함수는 반드시 선언되고 정의되어야 하며, main 함수에서 이 6개의 함수가 순차적으로 호출되어야 한다.

- 보고서는 **"assn5.doc"**, **"assn5.hwp"** 혹은 **"assn5.pdf"**로 저장 할 것.
- Assignment 문서에 명시된 경우를 제외하고 다른 예외처리를 할 필요는 없음.
- 문제의 출력 형식은 채점을 위해서 아래의 실행 예시와 최대한 비슷하게 작성 바람.

## (70 pts) 기본적인 설명 및 구현

### ■ 게임 설명

게임은 2인(딜러와 플레이어)의 게임으로 이루어지며, 컴퓨터가 딜러가 되고 사용자가 플레이어가 된다. 기본적으로 카드의 합이 21을 넘지 않는 범위 내에서 딜러와 숫자의 합을 겨루어 점수가 높으면 이기는 게임이다.

카드는 조커를 제외한 플레잉 카드 52장을 사용한다.

- 카드는 각 모양 4개 [♠ ♥ ♦ ♣]에 대하여 숫자 13개 [A 2 3 4 5 6 7 8 9 10 J Q K])가 쓰여진 총 52장의 카드를 사용한다. J, Q, K는 모두 숫자를 10으로 처리하며, A는 가진 사람에게 유리하게 1 혹은 11로 계산할 수 있다. 그 밖의 9 이하의 카드는 그 숫자대로 점수를 계산한다.

Ex> 플레이어의 카드가 A와 9라면, A를 11로 적용하여 플레이어의 카드 숫자 합은 20이다. 만약 플레이어의 카드가 A, K, 8이라면, A를 1로 적용하여 플레이어의 카드 숫자 합은 19이다. 이는 딜러에게도 그대로 적용된다.

게임의 방법은 다음과 같다.

- 플레이어는 카드를 받기 전에 베팅(Betting) 금액을 정하여 베팅한다.
- 딜러는 플레이어->딜러->플레이어->딜러 순으로 카드 1장씩을 나누어준다. 총 두 장씩의 카드를 받은 후, 카드를 공개한다. 이 때, 딜러의 첫 번째 카드는 공개하지 않고 덮어 놓는다.
- 처음 2장의 카드가 A(에이스)와 10(J, Q, K 포함)으로 21이 되는 것을 '블랙잭'이라고 하며, 플레이어가 블랙잭인 경우, 딜러의 카드와 관계없이 플레이어가 게임에서 승리하게 된다.
- 플레이어가 블랙잭이 아닌 경우, 플레이어는 자신의 카드의 합이 21에 가까워지도록 하기 위해서, 추가로 카드를 받을지 여부를 결정하게 된다. 카드는 1장씩 몇 장이라도 요구할 수 있으며, 카드를 더 받지 않는 것이 유리하다고 판단하면 추가하지 않아도 된다. (플레이어가 21을 넘은 경우, 게임에서 지며 베팅한 금액을 잃게 된다.)
- 플레이어의 카드 추가가 끝나면, 딜러는 덮어 놓은 카드를 공개한 후, 카드를 추가할 것인지 여부를 결정하게 된다. 딜러의 경우, 가진 카드의 총합이 16이하이면 무조건 카드를 계속 추가로 받아야 한다.
- 딜러의 카드 추가가 끝나면, 승패를 판정한다. 기본적으로 카드의 합이 21을 넘지 않는

범위에서 플레이어의 카드 합이 딜러 보다 높거나 같으면 승리하고 낮으면 패배한다.

- 배당율은 베팅한 금액만큼을 받는 것이 기본이므로, 플레이어가 이길 경우 베팅한 금액의 2배를 받고, 지면 잃는다.

## ■ 기본규칙 및 용어

- **블랙잭(Blackjack)** - 처음 받은 두 장의 카드의 합이 21이 되는 경우를 의미한다. 플레이어가 블랙잭인 경우 무조건 게임에서 승리한다.
- **히트(Hit)** - 카드를 1장 더 받겠다는 의사 표시이다. 가지고 있는 카드의 합이 21 이상이 아니면 원하는 만큼 Hit을 할 수 있다.
- **스테이(Stay)** - 카드를 더 이상 받지 않고 자신의 차례를 끝내는 것을 의미한다. Hit을 했을 때 합이 21이 될 경우 자동적으로 Stay가 된다. (Stand라고 하기도 한다.)
- **버스트(Bust)** - 가지고 있는 카드의 총합이 21을 넘는 경우를 의미한다. 이 경우, Hit을 할 수 없고, 플레이어가 Bust가 되면 패배한다. 플레이어가 Bust가 아닌 상황에서 딜러가 Bust가 되면 플레이어가 게임에서 이긴 것이 된다.

## ■ 프로그램 진행

프로그램은 다음과 같은 순서로 진행된다.

프로그램 실행 → 게임시작 → 1. 베팅 → 2. 카드 섞기(셔플링) → 3. 카드 분배 → 4.

플레이어 단계 수행 → 5. 딜러 단계 수행 → 6. 승패 판정 → 새로운 게임 시작

## ■ 구현

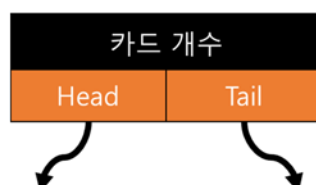
52장의 플레잉 카드에서 딜러와 플레이어가 카드를 받고 진행하는 게임이기 때문에 카드와 관련된 모든 것들은 구조체와 Linked List로 관리되어야 한다.

게임에 사용할 카드(링크드 리스트의 노드에 해당)에 대한 정의는 구조체로 작성한다. 아래 그림처럼, 기본적으로 카드의 그림 모양과 숫자를 나타내는 정보와 다른 카드와 연결이 되는 포인터를 반드시 포함해야 한다. 이외의 필요한 정보는 추가적으로 정의할 수 있다.

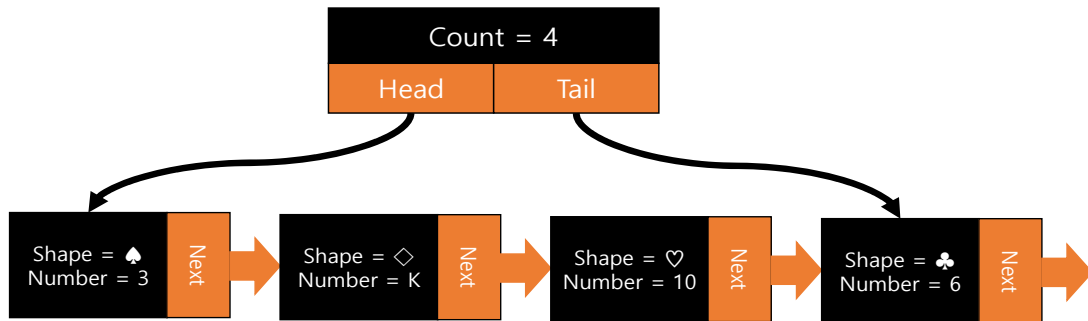


플레이어는 플레이어 자신의 카드 세트(set)을 가지고 있고, 딜러 또한 딜러의 카드 세트를 가지고 있다. 아직 플레이어와 딜러에게 가지 않은 카드 세트를 덱(deck)이라고 칭한다.

카드 세트는 다음 그림과 같이 카드 개수, Head, Tail 정보들을 반드시 포함시켜 구조체로 정의해야 하며, 이외에 필요한 정보를 추가적으로 정의할 수 있다. (Head와 Tail이 가리키는 것은 각각 첫 번째 노드와 마지막 노드이다.)



카드와 카드세트를 이용하여 Linked List를 만들면 아래와 같은 예시처럼 만들어진다.



## 1. 베팅 하기

- 처음에 들고 시작하는 소지금은 윈도우의 커맨드 라인(시작->실행->cmd)에서 명령줄인수(argc, argv)를 통해 받을 수 있게 구현되어야 한다. Argument는 하나 이하만을 받는다고 가정하며, 1 이상 1,000,000 이하의 자연수만 들어온다고 가정한다.

```
C:\Windows\system32\cmd.exe - assn5.exe 12200
C:\Users\GA-Mincheol Kim\Desktop>assn5.exe 12200
[현재 소지금 : $12200] 베팅 금액을 설정하세요! <0 for Exit>
```

- 추가적인 숫자를 입력하지 않은 경우, 처음 소지금은 \$5,000이다.

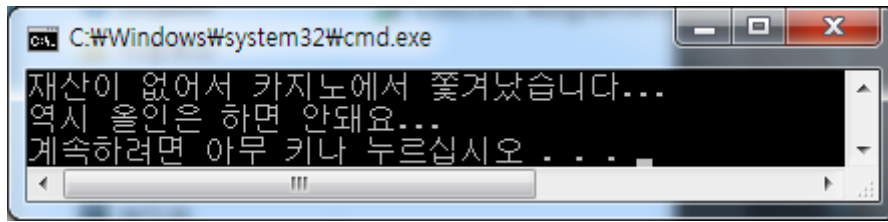
```
C:\Windows\system32\cmd.exe - assn5.exe
C:\Users\GA-Mincheol Kim\Desktop>assn5.exe
[현재 소지금 : $5000] 베팅 금액을 설정하세요! <0 for Exit>
```

- 베팅 금액을 설정한다. 베팅 금액은 키보드로 입력 받으며, 현재 가지고 있는 돈 이하의 0 이상의 정수로 들어와야 한다. 이 값은 int형 범위를 벗어나지 않는다고 가정한다.

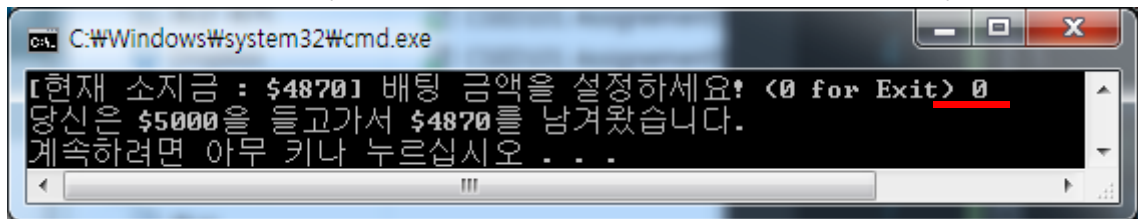
- 베팅 전, 소지금(게임 머니)이 처음 가지고 있던 소지금의 100배 이상이 된다면, 플레이어가 돈을 너무 많이 벌었으므로 프로그램을 종료한다. 종료 시의 문구는 자유롭게 넣되, 최초의 소지금과 현재 지니고 있는 금액을 출력해야 한다.

```
C:\Windows\system32\cmd.exe
당신은 돈을 너무 많이 벌어 카지노 업체가 당신을 쫓아냈습니다.
카지노에서 $5000으로 무려 $800219를 벌었습니다!!!
계속하려면 아무 키나 누르십시오 . . .
```

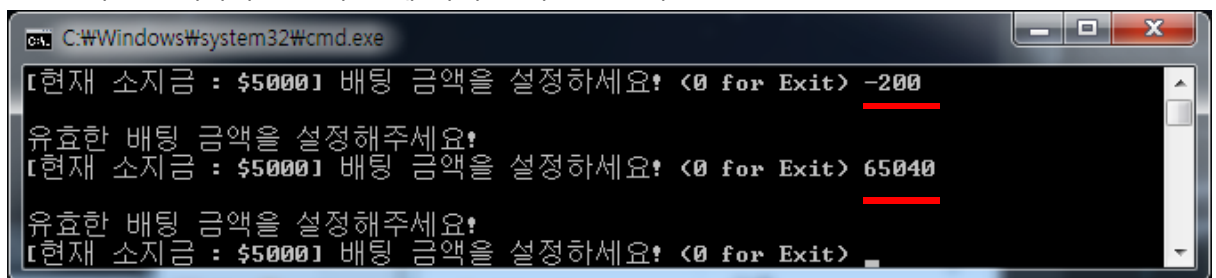
- B. 베팅 전, 소지금이 \$0 이라면, 돈이 없으니 카지노에서 쫓겨나야 하므로 프로그램을 종료한다. 도박은 위험하다는 문구를 꼭 넣어주자.



- C. 베팅 금액으로 0이 입력되면, 최초의 소지금과 현재 남아있는 금액을 출력하고 프로그램을 종료한다. (실행예시의 빨간 밑줄은 사용자 입력 부분에 해당)



- D. 베팅 금액으로 음수가 입력되거나 현재 소지금보다 많은 액수가 입력된 경우, 유효한 금액을 입력하라는 말과 함께 다시 입력을 받는다.



## 2. 카드 덱과 섞기

이 과제에서 구현하는 블랙잭은 한 벌(52장)의 플레잉 카드만을 사용한다. 카드 덱은 linked list를 이용하여 구현한다.

카드를 아래의 2가지 경우에 섞는다.

- (1) 게임을 처음 시작할 때 즉, 프로그램을 시작해서 첫 번째 게임을 시작하는 경우에 새로운 카드 52장을 섞어 사용한다.
- (2) 베팅 후, 게임 시작 전에 만약 덱의 카드 개수가 52장의 절반인 26장 이하가 된다면, 덱의 카드를 모두 버리고, 새로운 카드 52장을 섞어 사용한다.
  - 메모리 관리를 효율적으로 한다. 카드를 버리는 경우, 할당 받은 메모리를 할당 해제해야 한다.

이 경우, “카드를 새롭게 섞습니다...”라는 문구를 출력한 후, 아래의 화면처럼 사용자 입력을 기다린다. 엔터키를 입력받으면 게임이 시작된다. 이 때, 화면 `system("cls");` 를 사용하여 화면을 지운 후, 다음 단계(3. 카드 분배 화면)의 화면이 출력되도록 한다.

```
C:\Windows\system32\cmd.exe
[현재 소지금 : $5000] 배팅 금액을 설정하세요! <0 for Exit> 3500
카드를 새롭게 섞습니다...
```

### 3. 카드 분배

실제 게임이 시작되면, 플레이어와 딜러는 텍에 있는 카드를 받게 된다. 이 때, 카드는 플레이어->딜러->플레이어->딜러 순으로 한 장씩 총 2장을 나누어 준다.

이 때의 출력 화면에서는 아래와 같다.

```
C:\Windows\system32\cmd.exe
[소지금 : 1500] [배팅금액 : 3500]
-----
<Dealer> [????] [♦ 9] , Sum = 9
<Player> [♣ Q] [♦ 3] , Sum = 13
Hit or Stay?
```

첫 번째 줄은 플레이어가 베팅하고 남은 금액(베팅 금액을 제외한 액수)과 베팅 금액이 출력된다. 세 번째 줄은 딜러의 카드 리스트(첫 카드의 정보는 숨겨짐)와 카드 숫자의 총합(첫 카드를 제외한 합)을 출력한다. 그 다음 줄은 플레이어의 카드 리스트와 카드 숫자의 총합을 출력하고 출력한다.

Hit or Stay? 메시지는 플레이어 단계로 넘어가게 되는 경우 출력 된다.

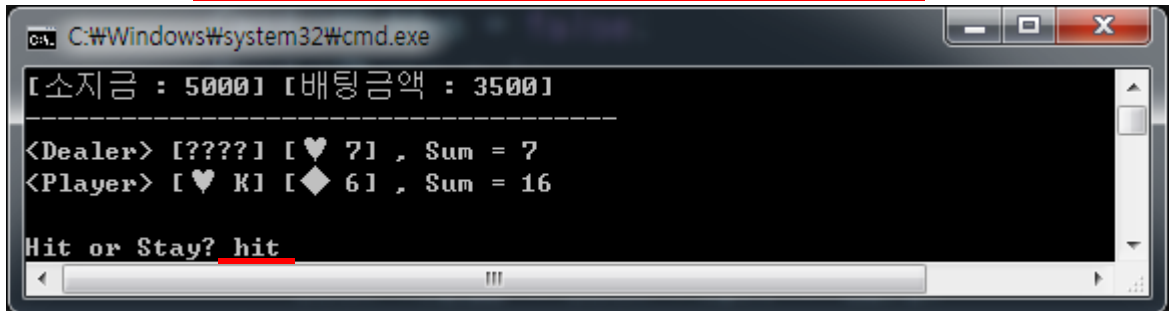
이 순서에서 플레이어의 카드 합이 21이 되어 Blackjack이 된 경우는 플레이어의 승리로 베팅한 금액의 3배를 얻게 되며, 게임이 끝난다. 게임이 종료되었으니, 베팅 금액을 입력 받는 화면으로 넘어간다.

Ex> 현재 소지금이 \$1,500이고, \$3,500을 베팅 금액으로 설정했을 때, Blackjack이라면 베팅 금액의 3배를 획득하여 게임에서 얻은 금액은 \$10,500이 되고, 이에 따라 총 소지금은 \$12,000이 된다.

```
C:\Windows\system32\cmd.exe
[소지금 : 11800] [배팅금액 : 400]
-----
<Dealer> [????] [♣ 5] , Sum = 5
<Player> [♣ Q] [♣ A] , Sum = 21
★B★L★A★C★K★J★A★C★K★
배팅한 금액의 3배에 해당하는 $1200를 획득했습니다!
```

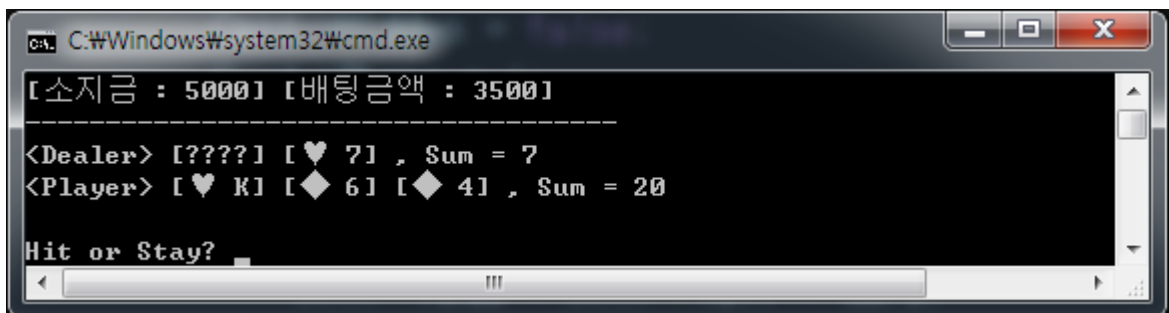
#### 4. 플레이어 단계

- 이 단계에서 플레이어는 카드를 더 받을지(Hit), 받지 않을 지(Stay)를 결정할 수 있다. 카드를 추가로 받는 경우에는 Hit을 입력, 받지 않을 경우에는 Stay를 입력한다.



```
C:\Windows\system32\cmd.exe
[소지금 : 5000] [배팅금액 : 3500]
-----
<Dealer> [????] [♥ 7] , Sum = 7
<Player> [♥ K] [♦ 6] , Sum = 16
Hit or Stay? hit
```

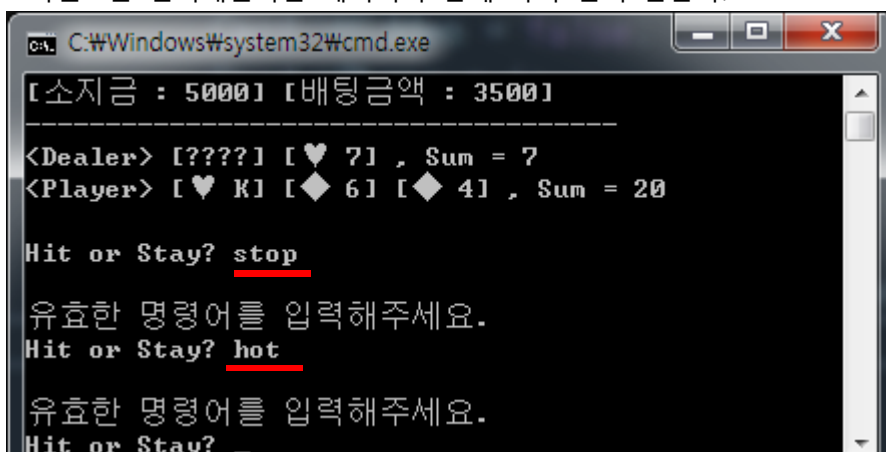
아래의 예시는, 위의 예시에서 hit를 입력하여 카드를 추가로 받은 경우로 플레이어의 카드 리스트 및 카드 합이 갱신 된 것을 볼 수 있다. 화면 출력 시, 화면을 지우고 아래와 같이 출력한다.



```
C:\Windows\system32\cmd.exe
[소지금 : 5000] [배팅금액 : 3500]
-----
<Dealer> [????] [♥ 7] , Sum = 7
<Player> [♥ K] [♦ 6] [♦ 4] , Sum = 20
Hit or Stay? 
```

- Hit을 입력**하여 카드를 추가로 받은 경우,
  - A. 카드의 합이 21보다 작을 때에는 플레이어 단계를 다시 진행하여 카드를 더 받을지를 물어본다.
  - B. 카드의 합이 21이상이라면 자동으로 딜러의 단계로 넘어간다.
- Stay를 입력**한 경우, 딜러 단계로 넘어간다.
- 입력 처리**

플레이어는 Hit, Stay를 입력 할 수 있다. 이들 입력은 대소문자와 상관없이 인식한다. 예를 들어, hit, Hit, HIT 등은 같은 의미로 처리한다. 그 이외의 입력에 대해서는 유효한 커맨드를 입력해달라는 메시지와 함께 다시 입력 받는다.



```
C:\Windows\system32\cmd.exe
[소지금 : 5000] [배팅금액 : 3500]
-----
<Dealer> [????] [♥ 7] , Sum = 7
<Player> [♥ K] [♦ 6] [♦ 4] , Sum = 20
Hit or Stay? stop
유효한 명령어를 입력해주세요.
Hit or Stay? hot
유효한 명령어를 입력해주세요.
Hit or Stay? 
```

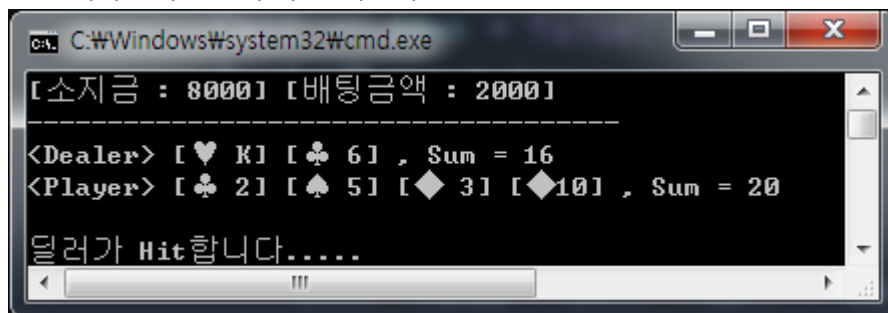
문자열은 20자를 넘지 않는다고 가정하자. (테스트 할 문자열 포함)



## 5. 딜러 단계

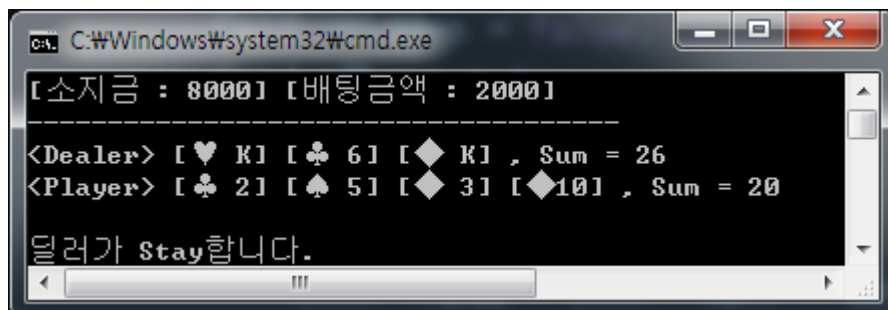
- 이 단계에서 딜러는, 정보를 숨겨 놓은 첫 카드를 포함한 카드 리스트와 카드 숫자의 합을 출력한다.
- 딜러는 정해진 룰에 따라 Hit과 Stay를 하게 되는데, 자신의 카드 숫자 합이 16이하면 무조건 Hit을 수행하고, 17이상이면 무조건 Stay를 수행하여 카드를 더 받지 않고 승패 판정 단계로 넘어간다.
- 플레이어가 blackjack인 경우를 제외하고, 딜러 단계는 반드시 진행된다.

아래는 딜러의 카드의 합이 16이하이므로 Hit을 수행하는 화면으로 아래에 "딜러가 Hit합니다..."라는 문구가 출력된다.



```
C:\Windows\system32\cmd.exe
[소지금 : 8000] [배팅금액 : 2000]
-----
<Dealer> [♥ K] [♣ 6] , Sum = 16
<Player> [♣ 2] [♠ 5] [♦ 3] [♦ 10] , Sum = 20
딜러가 Hit합니다.....
```

사용자가 엔터 입력시, 다음 단계(화면)로 넘어간다. 아래의 예시는 위 예시에서 Hit 수행 후, 엔터를 입력한 경우로, 딜러의 카드가 추가된 것을 확인할 수 있다. 카드의 합이 17이상이므로 딜러는 Stay를 수행하게 된다.



```
C:\Windows\system32\cmd.exe
[소지금 : 8000] [배팅금액 : 2000]
-----
<Dealer> [♥ K] [♣ 6] [♦ K] , Sum = 26
<Player> [♣ 2] [♠ 5] [♦ 3] [♦ 10] , Sum = 20
딜러가 Stay합니다.
```

## 6. 승패 판정

- 딜러의 순서가 끝나면 아래 승패 조건을 기준으로 승패를 판정한다.
- 승패의 조건은 아래와 같다.
  - A. 플레이어가 Bust 상태라면 플레이어가 Bust라는 문구와 함께 무조건 패배한다. A 조건에 따라 플레이어가 패하면 B 조건에 대한 것은 출력하지 않는다.
  - B. 딜러가 Blackjack이라면 딜러가 Blackjack이라는 말과 함께 플레이어가 패배한다. (플레이어가 Blackjack인 경우는 4단계에서 이미 승패가 결정된다.)
  - C. 플레이어가 Bust 상태가 아닌데 딜러가 Bust 상태라면 플레이어가 승리한다.
  - D. 참가자들(플레이어와 딜러)의 카드 합이 21이하인 경우에는, 플레이어의 카드 숫자 합이 딜러의 카드 숫자 합보다 크거나 같으면 플레이어가 승리한다.

- 베팅 금액 정산

- A. 플레이어가 승리한 경우, 베팅한 금액만큼의 돈을 추가로 더 얻는다.
- B. 플레이어가 패배한 경우, 베팅한 금액을 잃는다.

- 베팅 금액 정산이 끝나고, 결과가 출력된 후, 엔터키를 입력하면 베팅 금액 설정(1. 베팅 하기)부분으로 넘어간다. 게임을 다시 시작하게 되면, 플레이어와 딜러가 이전 게임에서 받은 카드를 모두 버려야 한다. 이 때 메모리 할당해제를 반드시 수행해야 하며, 메모리 관리를 효율적으로 하도록 하자.

Ex> 플레이어가 Bust 인 경우 (A 조건)

```

C:\Windows\system32\cmd.exe
[소지금 : 11000] [배팅금액 : 1000]
-----
<Dealer> [♠ J] [♥ 9] , Sum = 19
<Player> [♣ 8] [♥ 8] [♦ 4] [♣ 5] , Sum = 25

딜러가 Stay합니다.
-----
버스트! 플레이어가 졌습니다. 배팅한 돈을 잃었습니다.
  
```

Ex> 딜러가 Blackjack인 경우 (B 조건)

```

C:\Windows\system32\cmd.exe
[소지금 : 18200] [배팅금액 : 1000]
-----
<Dealer> [♦ A] [♠ J] , Sum = 21
<Player> [♦ 8] [♦ 2] , Sum = 10

딜러가 Stay합니다.
-----
딜러가 ★B★L★A★C★K★J★A★C★K★
배팅한 돈을 잃었습니다.
  
```

Ex> 딜러가 Bust 인 경우 (C 조건) (단, 플레이어는 Bust 가 아님)

```

C:\Windows\system32\cmd.exe
[소지금 : 8000] [배팅금액 : 2000]
-----
<Dealer> [♥ K] [♣ 6] [♦ K] , Sum = 26
<Player> [♣ 2] [♠ 5] [♦ 3] [♦ 10] , Sum = 20

딜러가 Stay합니다.
-----
플레이어가 이겼습니다! $4000를 획득했습니다.
  
```

Ex> 플레이어와 딜러의 각각 카드의 합이 21이하인 경우 (D 조건)

```
C:\Windows\system32\cmd.exe
[소지금 : 5000] [배팅금액 : 3500]
-----
<Dealer> [♠ J] [♥ 7] , Sum = 17
<Player> [♥ K] [♦ 6] [♦ 4] , Sum = 20

딜러가 Stay합니다.
-----
플레이어가 이겼습니다! $7000를 획득했습니다.
```

```
C:\Windows\system32\cmd.exe
[소지금 : 14800] [배팅금액 : 200]
-----
<Dealer> [♠ 9] [♠ K] , Sum = 19
<Player> [♦ 9] [♥ Q] , Sum = 19

딜러가 Stay합니다.
-----
플레이어가 이겼습니다! $400를 획득했습니다.
```

```
C:\Windows\system32\cmd.exe
[소지금 : 15000] [배팅금액 : 2000]
-----
<Dealer> [♦ Q] [♥ J] , Sum = 20
<Player> [♥ 4] [♠ 10] , Sum = 14

딜러가 Stay합니다.
-----
플레이어가 졌습니다. 배팅한 돈을 잃었습니다.
```

## 7. 프로그램 종료

- 배팅금액 설정(1. 베팅 하기) 시에 0을 입력하거나, 현재 소지금의 100배 이상이 되는 경우, 혹은 소지금의 \$0인 경우 프로그램을 종료한다.

### (문제 관련 중요사항 및 팁)

- 앞서 언급했듯이, 모든 카드 리스트들과 카드들은 struct Type과 Linked List를 통해서 관리가 되어야 한다. 위에서 언급한 정보들은 모두 그 Type에 들어 있어야 하며, 필요 시 새로운 정보들을 저장하는 변수 및 포인터를 추가 및 수정, 활용할 수 있다. 이렇게 추가된 정보에 대한 설명은 반드시 필요하다.
- 카드를 새로 섞는 것은 임의로 섞어야 한다. srand(time(NULL)) 구문을 이용하고, rand 함수를 이용한다. 다만 Blackjack의 경우를 확인하기 위해서는 그 룰에 맞는 조건이 불기 때문에 확인 한 번을 위해 여러 번 게임을 하게 되는 상황이 발생한다. 이럴 때는 랜덤하게 카드가 섞이는 부분은 주석처리를 하고 임의로 자신이 임의로 덱을 만들어서 하는 것도 하나의 방법이다. 물론 최종적으로 구현된 코드는 랜덤하게 카드를 섞어야 한다.
- 카드 리스트를 갱신할 때마다 화면을 깨끗하게 지우고 새로 출력할 수 있게 하는 system("cls") 문구를 사용하는 것을 추천하고, 딜러가 Hit을 할 때는 그 정보가 순차적으로 보여야 하므로 getch(), system("pause")를 사용할 것을 추천한다. system 함수의 경우 <stdlib.h>에 구현이 되어 있다.
- 이 부분의 핵심은 J, Q, K를 10으로 보는 것과 A를 현재 카드 리스트를 지닌 딜러나 플레이어에게 각각 유리하게 적용하는 데에 있다. 11과 1의 차이는 10이라는 것에 유념하고, A의 개수를 세어 주는 것이 따로 있다면 편할 것이다.
- 구현을 하다 보면 타입이 같고 이름이 다른 변수나 포인터들이 하는 동작이 중복되는 경우가 상당히 많은데, 이런 경우에 함수를 써서 한번에 관리하는 것이 편리하다.
- 주석이나 printf 문구를 영어로 쓰거나 한글로 쓰거나 하는 것은 자유이다.