CSE2011 Problem Solving, 2016 Spring

2013312343 이상헌

**Homework 2-2**

**1. 문제 이해**

(1) 문제

A Foker deck contains 32 cards. Each card has a suit of either clubs, diamonds, hearts, or spades (denoted C, D, H, S in the input data). Each card also has a value of either 2 through 9 (denoted 2, 3, 4, 5, 6, 7, 8, 9). For scoring purposes card values are ordered as above, with 2 having thelowest and 9 the highest value. The suit has no impact on value.

A Foker hand consists of three cards dealt from the deck. Foker hands are ranked by the following partial order from lowest to highest.

 High Card. Hands which do not fit any higher categoryare ranked by the value of their highest card. If the highest cards have the same value, the hands are ranked by the next highest, and so on. (Suits do not affect the rank.)

 Pair. Two of the three cards in the hand have the same value. Hands which both contain a pair are ranked by the value of the cards forming the pair. If these values are the same, the hands are ranked by the values of the cards not forming the pair, in decreasing order.

 Three of a Kind. All the three cards in the hand have the same value. Hands which both contain three of a kind are ranked by the value of the three cards.

 Flush. Hand contains three cards of the same suit. Hands which are both flushes are ranked using the rules for High Card.

 Straight. Hand contains three cards with consecutive values. Hands which both contain a straight are ranked by their highest card.

 Straight Flush. Three cards of the same suit with consecutive values. Ranked by the highest card in the hand.

To compare several pairs of Foker hands and to indicate which, if either, has a higher rank. Each player has three cards, but each player can change the value of one card (to any value) so as to make the player’s Foker hand as high as possible.

Input: The input file contains several lines, each containing the designation of six cards: the first three cards are the hand for the player named “Black” and the next three cards are the hand for the player named “White”.

Output: For each line of input, print a line containingone of the following:

(2) 중요 정보

- 각 카드는 2~9의 숫자와 C, D, H ,S의 문양을 갖는다.

- 두 명의 플레이어는 각각 3장의 카드를 받는다.

- 3장의 카드의 조합은 주어진 족보에 의거하여 우선순위를 정하게 한다.

- 각 플레이어는 자신의 보유한 카드 패의 우선순위를 높이기 위하여 3장의 카드 중에 한 장의 카드의 숫자를 임의로 바꿀 수 있다.

(3) 문제 정의

- 주어진 카드를 받은 두 명의 플레이어에 대해, 승부를 결정하는 프로그램을 작성한다.

**2. 문제 해결**

(1) 해결 방법 결정

*① 가장 높은 우선순위의 족보부터.*

- 카드 패의 우선순위를 결정하는 족보는 high card, pair, three of a kind, flush, straight, straight flush가 있다. 이들 중에서 가장 높은 우선순위를 결정하는 straight flush부터 거꾸로 생각해보기로 하였다. 또한 플레이어가 한 카드의 숫자를 바꿀 수 있다는 점을 크게 고려하였다.

- 아이디어

플레이어는 자신의 마음에 들지 않는 카드 한 장의 숫자를 임의로 바꿀 수 있다. 결국 우선순위를 결정하는 것은 세 카드 중에 두 카드이다. 가장 높은 족보부터 그 족보가 나타날 수 있는 경우를 모두 세어보았다. 한 카드의 숫자를 바꾸어 만든, 우선순위가 가장 높은 패의 족보는 숫자로 나타내어, 나중에 승부를 겨룰 때에 보다 쉽게 비교할 수 있도록 한다. (족보 점수라 칭하였다.)

1. Straight Flush와 Straight

Straight Flush는 세 카드의 문양이 같으면서도, 숫자가 연속 되어야 한다. Straight는 문양에 상관 없어 숫자만 연속적이면 된다. 세 장의 카드가 straight가 되는 경우는 크게 두 가지이다. 세 장 중에 두 장이 연속적이거나, 세 장 중에 두 장의 차이가 2인 경우이다.

첫 번째의 경우에는 나머지 한 장의 카드를 두 장에 이어 연속한 수로 바꾸면 가능하다. 예를 들어, 8H 5H 4D의 패를 받은 경우, 5H와 4D가 연속적이므로 8H의 8을 나머지 두 수에 이어 연속한 수로 바꾼다면 (6 또는 3이 될 것이다.) Straight를 만들 수 있다. 연속한 수는 보통 두 가지로 나타나는데, 같은 Straight 내에서는 가장 큰 수가 큰 패가 유리하므로, 연속한 수 중 큰 수로 바꾼다.

만일 두 장이 연속하지만 한 장의 카드의 숫자가 9인 경우, 나머지 한 장의 카드를 연속하면서도 큰 수로 잡을 수 없다. 예를 들어 9H 8D 3D라는 카드에서는 3을 9, 8과 연속한 수로 바꾸어 Straight를 만들 수 있는데, 9보다 큰 수로 바꾸지는 못하므로 이런 경우에는 3을 연속한 수 중 작은 수인 7로 바꾼다.

두 번째의 경우에는 나머지 한 장의 카드를 두 장 사이의 수로 바꾸면 Straight가 가능하다. 예를 들어 8H 6D 2S의 패를 받은 경우, 2를 8과 6 사이의 숫자인 7로 바꾼다면 Straight를 만들 수 있다.

Straight Flush는 숫자를 바꾸어 만든 패가 Straight인 경우에 대해 추가적으로, 카드의 문양을 검사하여 알아낼 수 있다. 만일 Straight를 이루는 패의 카드의 문양이 모두 같다면, 이는 Straight Flush가 된다.

Straight Flush는 족보 점수 52~59점을 갖고, Straight는 족보 점수 42~49점을 갖는다. 일의 자리는 가장 높은 카드의 숫자를 의미한다.

1. Flush

Flush는 세 카드의 문양이 모두 같은 경우이다. 문양은 플레이어가 임의로 바꿀 수 없기 때문에, 받은 세 카드의 문양을 그대로 조사한다. 만일 세 카드가 같은 문양을 가지고 있다면, Flush가 된다. 만일 두 플레이어가 모두 Flush라면, 세 카드의 숫자 중에 가장 높은 숫자를 비교하여 더 높은 숫자를 가지고 있는 플레이어가 승리한다. 그러므로 한 장의 카드의 숫자를 바꾸는 기회는 아무 숫자를 9로 바꾸는 것으로 한다. 9 Flush는 Flush 족보 중에 가장 높은 우선순위를 갖기 때문이다.

Flush의 족보 점수는 32~39점을 갖고, 일의 자리는 가장 높은 카드의 숫자를 의미한다. 하지만 플레이어가 임의로 한 카드의 숫자를 바꾸는 기회가 있기 때문에 모든 Flush의 족보 점수는 39점이 될 것이다.

1. Three of a kind

Three of a kind는 세 카드의 숫자가 같은 경우이다. 플레이어는 패를 받은 후에 한 장의 카드의 숫자를 교환할 기회가 있기 때문에, 만일 두 장의 카드의 숫자가 같다면, 나머지 한 장의 숫자를 바꾸어 Three of a kind 족보를 만들 수 있다.

Three of a kind의 족보 점수는 22~29점을 갖고, 일의 자리는 카드의 숫자를 의미한다.

1. Pair

Pair는 세 장 중에 두 장의 카드의 숫자가 같은 경우이다. 한 장의 카드의 숫자를 바꾸는 기회를 고려한다면, 임의의 세 장의 카드를 받아도 pair 족보를 만들 수 있다. 그러므로 세 장의 카드 중에서 가장 큰 숫자를 갖는 카드로 맞추어 pair 족보를 만든다.

Pair의 족보 점수는 12~19점을 갖고, 일의 자리는 pair를 이루는 숫자를 의미한다. 만일 두 플레이어가 똑같은 숫자의 카드로 pair 족보를 가지고 있다면, 승부는 pair를 이루지 않는 나머지 한 장의 숫자에 따라 결정된다. 나머지 카드의 숫자가 클수록 족보의 우선순위가 높아진다. (in decreasing order) 이는 족보 점수에 반영하지는 않았지만, 승부를 결정할 때에 고려할 사항이다.

- 이점

족보를 거꾸로 생각하는 것은, 한 장의 카드의 숫자를 바꾸는 단계에서 가능한 우선순위가 높은 족보로 바꾼다는 것을 의미한다. 우선순위가 가장 높은 족보부터 생각하면, 후에 그보다 우선순위가 낮은 족보로 교환 가능해도 바꾸지 않는다. 또한 족보 점수라는 아이디어는 승부의 결과를 보다 쉽게 판단할 수 있고, 같은 족보 상에서 숫자가 높은 카드를 갖는 플레이어가 승리한다는 규칙을 보다 쉽게 반영할 수 있다.

**3. 코드 작성**

- 본 아이디어 및 설명을 바탕으로 코드를 작성하였다.

#include <stdio.h>

/\* You can modify from here \*/

int Foker\_hands(char BN1, char BS1, char BN2, char BS2, char BN3, char BS3, char WN1, char WS1, char WN2, char WS2, char WN3, char WS3) {

int Bvalue = 0, Wvalue = 0;

char bnl, bnm, bns, wnl, wnm, wns;

Bvalue, Wvalue는 각각 black과 white 플레이어의 족보 점수이다. 0으로 초기화한다. Bnl, bnm, bns, wnl, wnm, wns는 각각의 플레이어가 받은 세 장의 카드의 숫자를 내림차순으로 정리할 필요성을 느끼고, 저장한 character 변수이다. (각각의 l : lagre, m : medium, s : small을 의미하고, bn : black number, wn : white number를 의미한다.)

if (BN1 >= BN2) {

if (BN1 >= BN3) {

bnl = BN1;

if (BN3 >= BN2) {

bnm = BN3;

bns = BN2;

}

else {

bnm = BN2;

bns = BN3;

}

}

else {

bnl = BN3;

bnm = BN1;

bns = BN2;

}

}

else {

if (BN2 >= BN3) {

bnl = BN2;

if (BN3 >= BN1) {

bnm = BN3;

bns = BN1;

}

else {

bnm = BN1;

bns = BN3;

}

}

else {

bnl = BN3;

bnm = BN2;

bns = BN1;

}

}

Black 플레이어가 받은 세 장의 카드의 숫자를 내림차순으로 정리하여, 가장 큰 수를 bnl에, 중간 값을 bnm에, 가장 작은 수를 bns에 저장한다.

if (WN1 >= WN2) {

if (WN1 >= WN3) {

wnl = WN1;

if (WN3 >= WN2) {

wnm = WN3;

wns = WN2;

}

else {

wnm = WN2;

wns = WN3;

}

}

else {

wnl = WN3;

wnm = WN1;

wns = WN2;

}

}

else {

if (WN2 >= WN3) {

wnl = WN2;

if (WN3 >= WN1) {

wnm = WN3;

wns = WN1;

}

else {

wnm = WN1;

wns = WN3;

}

}

else {

wnl = WN3;

wnm = WN2;

wns = WN1;

}

}

White 플레이어가 받은 세 장의 카드의 숫자를 내림차순으로 정리하여, 가장 큰 수를 wnl에, 중간 값을 wnm에, 가장 작은 수를 wns에 저장한다.

//black 점수채점

if (bnl - bnm == 1 || bnl - bns == 1) {

if (bnl == '9') {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 59)

Bvalue = 59;

}

else {

if (Bvalue < 49)

Bvalue = 49;

}

}

else {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 50 + bnl - 48 + 1)

Bvalue = 50 + bnl - 48 + 1;

}

else {

if (Bvalue < 40 + bnl - 48 + 1)

Bvalue = 40 + bnl - 48 + 1;

}

}

}

Black 플레이어가 받은 카드의 족보를 결정한다. Straight가 되는 경우부터 생각한다. 만일 가장 큰 수가 중간 수 혹은 가장 작은 수보다 1만큼 더 크다면, Straight를 완성할 수 있다. 단, 가장 큰 수가 9인 경우엔 9 Straight가 되므로 따로 점수 계산을 한다. (따로 계산하는 이유는 밑에 설명되어 있다.) 추가하여 세 카드의 문양이 같다면 Straight Flush가 되어 52~59점의 점수를 얻는다.

족보 점수의 예시를 보면, Bvalue = 50 + bnl - 48 + 1 와 같이 나타내었다. Bnl은 character변수이므로, integer로 계산 시 실제 integer보다 48만큼 더 크다. (예를 들어, character 0은 integer로 변환 시 48이 된다.) 따라서 48을 빼준다. 1을 더한 것은 가장 높은 수가 9가 아니면서 Straight가 되는 경우인데, bns를 bnl보다 1 큰 수로 변환한다면, 1만큼 더 큰 점수를 얻을 수 있기 때문이다.

if (bnm - bns == 1) {

if (bnm == '9') {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 59)

Bvalue = 59;

}

else {

if (Bvalue < 49)

Bvalue = 49;

}

}

else {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 50 + bnm - 48 + 1)

Bvalue = 50 + bnm - 48 + 1;

}

else {

if (Bvalue < 40 + bnm - 48 + 1)

Bvalue = 40 + bnm - 48 + 1;

}

}

}

만일 중간 수가 가장 작은 수보다 1만큼 더 크다면, Straight를 완성할 수 있다. 단, 중간 수가 9인 경우엔 9 Straight가 되므로 점수 계산시에 1을 더하지 않으므로 따로 계산한다. 추가하여 세 카드의 문양이 같다면 Straight Flush가 되어 52~59점의 점수를 얻는다.

if (bnl - bnm == 2 || bnl - bns == 2) {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 50 + bnl - 48)

Bvalue = 50 + bnl - 48;

}

else {

if (Bvalue < 40 + bnl - 48)

Bvalue = 40 + bnl - 48;

}

}

if (bnm - bns == 2) {

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 50 + bnm - 48)

Bvalue = 50 + bnm - 48;

}

else {

if (Bvalue < 40 + bnm - 48)

Bvalue = 40 + bnm - 48;

}

}

만일 가장 큰 수가 중간 수 혹은 가장 작은 수보다 2만큼 더 크다면, Straight를 완성할 수 있다. 이 경우에는 나머지 한 장의 카드를 두 숫자의 중간 숫자로 변환하기 때문에 가장 큰 수가 9가 포함되는 경우인지를 따로 고려할 필요가 없다. 추가하여 세 카드의 문양이 같다면 Straight Flush가 되어 52~59점의 점수를 얻는다.

if (BS1 == BS2 && BS2 == BS3) {

if (Bvalue < 39)

Bvalue = 39;

}

두 번째로 Flush가 될 수 있는지를 조사한다. 이는 카드의 숫자에 상관 없이 문양이 같아야 하므로, 카드의 문양만 조사한다. 만일 세 카드의 문양이 같다면 Flush가 되며, 한 카드의 숫자를 바꾸어 9 Flush로 만들 수 있다. 족보 점수는 39점이 된다.

if (bnl == bnm || bnm == bns) {

if (Bvalue < 20 + bnm - 48)

Bvalue = 20 + bnm - 48;

}

세 번째로 Three of a kind가 될 수 있는지를 조사한다. 이는 세 장 중에 두 장의 숫자가 같은 경우에, 나머지 한 장의 숫자를 바꾸면서 가능하다. 족보 점수는 22~29점으로 세 장의 숫자를 일의 자리로 한다. 세 장 중에 두 장이 같다면 그 두 장 중에는 항상 bnm이 존재하므로, 족보 점수는 bnm을 사용하여 계산한다.

if (Bvalue == 0) {

Bvalue = 10 + bnl - 48;

}

마지막으로 pair가 될 수 있는지를 조사한다. 임의의 세 장의 카드를 받은 경우, 한 장의 카드의 숫자를 바꿀 수 있는 기회가 제공되기 때문에 항상 pair를 만들 수 있다. 세 장 중에 가장 높은 카드로 pair를 만드는 것이 가장 높은 우선순위를 나타내기 때문에, 위 코드와 같이 족보 점수를 계산한다.

// white 점수 채점

White의 족보를 결정하는 것은 Black과 동일하다. 변수만 bnl에서 wnl로, bnm에서 wnm으로, bns에서 wns로 변한다.

if (wnl - wnm == 1 || wnl - wns == 1) {

if (wnl == '9') {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 59)

Wvalue = 59;

}

else {

if (Wvalue < 49)

Wvalue = 49;

}

}

else {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 50 + wnl - 48 + 1)

Wvalue = 50 + wnl - 48 + 1;

}

else {

if (Wvalue < 40 + wnl - 48 + 1)

Wvalue = 40 + wnl - 48 + 1;

}

}

}

if (wnm - wns == 1) {

if (wnm == '9') {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 59)

Wvalue = 59;

}

else {

if (Wvalue < 49)

Wvalue = 49;

}

}

else {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 50 + wnm - 48 + 1)

Wvalue = 50 + wnm - 48 + 1;

}

else {

if (Wvalue < 40 + wnm - 48 + 1)

Wvalue = 40 + wnm - 48 + 1;

}

}

}

if (wnl - wnm == 2 || wnl - wns == 2) {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 50 + wnl - 48)

Wvalue = 50 + wnl - 48;

}

else {

if (Wvalue < 40 + wnl - 48)

Wvalue = 40 + wnl - 48;

}

}

if (wnm - wns == 2) {

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 50 + wnm - 48)

Wvalue = 50 + wnm - 48;

}

else {

if (Wvalue < 40 + wnm - 48)

Wvalue = 40 + wnm - 48;

}

}

if (WS1 == WS2 && WS2 == WS3) {

if (Wvalue < 39)

Wvalue = 39;

}

if (wnl == wnm || wnm == wns) {

if (Wvalue < 20 + wnm - 48)

Wvalue = 20 + wnm - 48;

}

if (Wvalue == 0) {

Wvalue = 10 + wnl - 48;

}

// 승부 가르기

if (Bvalue > Wvalue)

return -1;

else if (Bvalue < Wvalue)

return 1;

else {

if (Bvalue >= 10 && Bvalue <= 19) {

if (bnm > wnm)

return -1;

else if (bnm < wnm)

return 1;

else

return 0;

}

else

return 0;

}

}

두 플레이어의 족보 점수를 이용하여 승부를 결정한다. Bvalue가 Wvalue보다 커서 Black이 승리하면 -1을, 작아서 White가 승리하면 1을 리턴 하고 함수를 종료한다. 족보 점수가 같은 경우, 두 가지 경우를 생각한다. 첫 번째로 pair로 같은 경우이다. Pair로 동급이 된다면 pair를 이루지 않은 나머지 한 장의 카드의 숫자가 큰 쪽이 승리하기 때문에, 두 플레이어가 가지고 있는 중간 숫자를 비교한다. Black이 높은 경우 black이 승리하여 -1을 리턴 하고, white가 높은 경우 white가 승리하여 1을 리턴 한다. 두 번째는 완전히 tie인 경우로, 0을 리턴 하고 함수를 종료한다.

**4. 확인 및 분석**

- 본 아이디어를 이용하여 프로그램을 작성하였다. 이 프로그램이 정확한 결과값을 도출하는지를 알아보기 위해서 간단한 여섯 장의 카드를 예시로 만들었다.

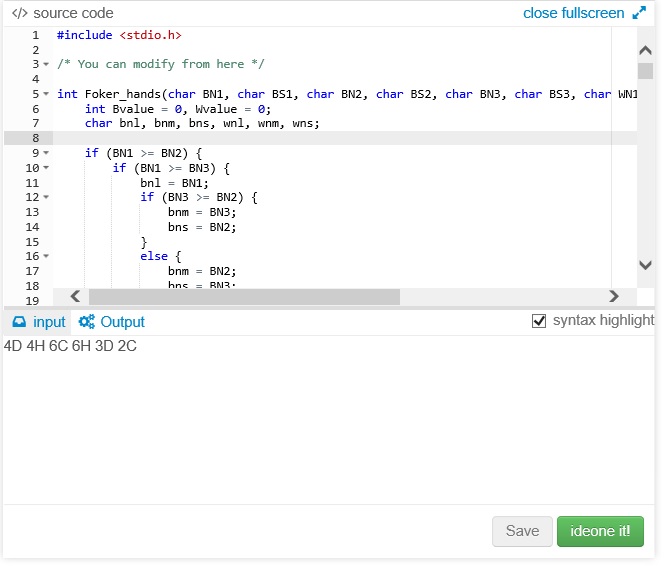


Figure . 예시 1 Input

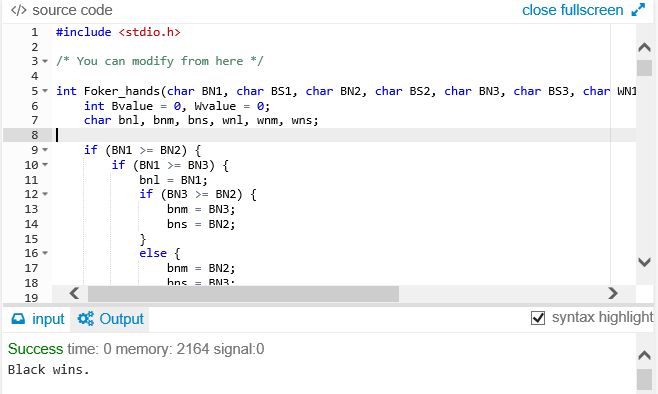


Figure . 예시 1 Output

- 예시 1은 Black에게 4D 4H 6C의 카드를, White에게 6H 3D 2C의 카드를 준다. Black은 4와 6의 차이가 2이므로 이를 바탕으로 Straight를 만들 수 있다. (모양이 달라서 Straight Flush는 되지 않는다.) 바뀐 Black의 패는 5D 4H 6C가 되어 46점의 족보 점수를 얻을 것이다. White는 3과 2의 차이가 1이므로 이를 바탕으로 Straight를 만들 수 있다. (모양이 달라서 Straight Flush는 되지 않는다.) 바뀐 White의 패는 4H 3D 2C가 되어 44점의 족보 점수를 얻을 것이다. 이에 따라 Black이 승리할 것이고, 프로그램은 Figure 2와 같이 정확한 결과를 나타내었다.

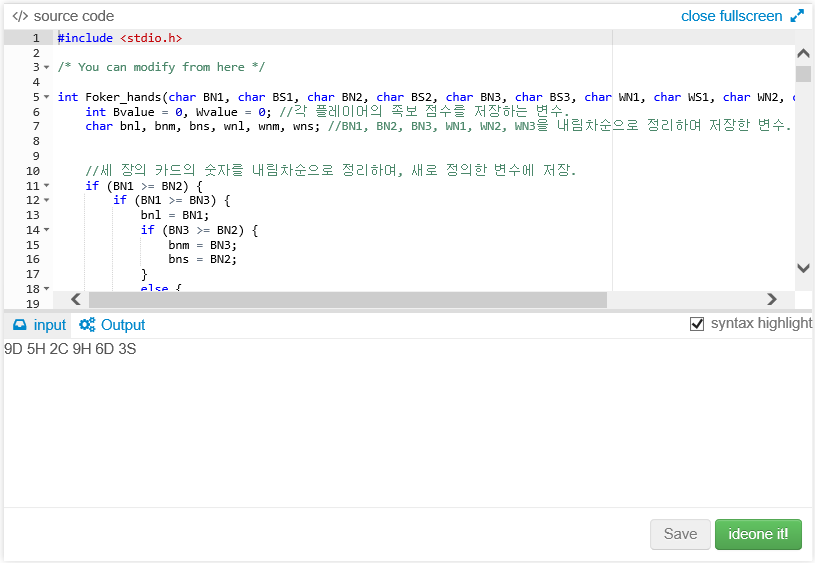


Figure . 예시 2 Input

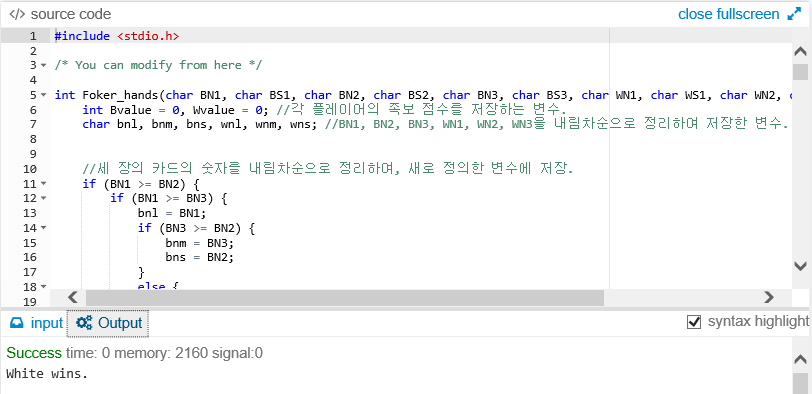


Figure . 예시 2 Output

- 예시 2는 Black에게 9D 5H 2C의 카드를, White에게 9H 6D 3S의 카드를 준다. 두 패는 모두 pair 이외의 다른 족보를 나타내지 못한다. 따라서 Black과 White 모두 19점의 족보 점수를 얻는다. 승부를 결정하는 과정에서 pair를 이루지 않은 나머지 한 장의 숫자의 크기를 비교한다. Black의 숫자는 5이고 White의 숫자는 6이므로, White의 숫자가 더 크기 때문에 White가 승리할 것이다. 프로그램은 Figure 4와 같이 정확한 결과를 나타내었다.

**6. 고찰**

- 본 문제는 두 명의 플레이어가 세 장의 카드를 받고, 족보를 결정하여 승부를 출력하는 프로그램이다. 높은 우선순위의 족보부터 생각하였다는 점과, 족보 점수라는 변수를 도입하는 것으로 본 프로그램을 구현하였다. 한 장의 카드의 숫자를 임의로 바꿀 수 있다는 점이 문제의 가장 큰 변수였지만, 세 장 중에 두 장만으로 결정할 수 있는 족보를 먼저 생각함으로써 나머지 한 장의 숫자를 모든 가능한 숫자로 보았다. 문제의 큰 변수가 오히려 힌트가 되었다고 생각한다.

**7. 참고문헌 및 사용**

- [www.ide.com](http://www.ide.com) : C언어 구현 및 실행.

- 이진규 교수님, [문제해결기법] CSE2011\_2016spring\_Lecture\_Note03, 2016