

# 과제 4

자료형이 자유로운 배열  
우리의마블  
도서관리

20232907 정현승

## 1. 문제 1

char형 50개만큼을 저장할 수 있는 배열에 char형, short형, int형, long long형 변수의 값을 저장한다. 이번에 입력받은 자료형의 종류는 변수 값 입력 전 입력받으며, 순서대로 1이 char형이고 4가 long long형이다. char형 배열에 저장할 수 있는 만큼 저장하고, 더 이상 저장할 수 없다면 저장 공간이 가득 찼다는 메시지를 출력한 뒤, 총 입력받은 숫자의 수와 각 변수의 종류, 값을 출력한다. char형 배열 이외의 변수는 5개를 넘을 수 없으며 다른 배열 또한 선언할 수 없다.

## 2. 문제 1 해결 방안

우선 입력과 출력 시 char 형 배열 내에서 현재 읽거나 쓰는 위치(커서)를 저장할 포인터 변수와, 입력받은 변수의 수를 저장할 변수 1개씩을 선언한다. 입력 시 scanf 함수를 이용해 이번 자료형의 종류를 입력받는다. 이 변수의 종류도 char 형 배열에 넣어야 하니 입력받은 변수는 바로 이번 저장 위치(커서 위치)에 저장하고 커서를 한 칸 뒤로 옮긴다. 그리고 배열의 남은 부분이 이번 변수를 입력받기에 충분한지 확인한 후, 입력받을 수 있는 경우 각 자료형에 맞는 scanf 서식을 이용해 저장 위치(커서)에 저장 후 이번 자료형의 크기만큼 커서를 뒤로 옮긴다. Char 형 배열의 시작 위치와 커서와의 차이값(즉 현재까지 사용한 공간)과 이번에 저장할 자료형의 크기가 배열의 크기인 50을 초과할 경우, 저장 공간이 없다는 메시지를 출력하고 입력을 종료한다.

출력 시 다시 커서를 배열 맨 앞으로 옮겨주고, 입력받을 때 따로 센 입력받은 숫자의 수를 출력한다. 이후 커서 위치에 저장된 이번 숫자의 자료형을 읽고 커서를 한 칸 뒤로 옮긴 뒤, 숫자를 출력하고 커서를 이번 자료형의 크기만큼 뒤로 옮긴다. 이를 입력받은 숫자의 수만큼 반복한 뒤 프로그램을 종료하여 문제를 해결했다.

이를 적용하여 문제 1을 해결한 결과는 다음과 같다.


### 3. 문제 1의 결과



```
Microsoft Visual Studio 디버그 콘솔
= 50바이트 남음
>1 15
= 48바이트 남음
>2 10000
= 45바이트 남음
>3 1931
= 40바이트 남음
>4 9382
= 31바이트 남음
>4 123456789012
= 22바이트 남음
>3 2100098987
= 17바이트 남음
>1 17
= 15바이트 남음
>1 65
= 13바이트 남음
>1 97
= 11바이트 남음
>1 32
= 9바이트 남음
>1 123
= 7바이트 남음
>2 132
= 4바이트 남음
>3 100
= 저장 공간이 가득 찼습니다.
12
15C 10000S 1931I 9382LL 123456789012LL 2100098987I 17C 65C 97C 32C 123C 132S
D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\중간고사\64\Debug\중간고사.exe(프로세스 14956개)이(가) 종료되었습니다(
코드: 0개).
```

그림 1

위 그림 1은 실행 예시를 그대로 입력한 것이다.



```
Microsoft Visual Studio 디버그 콘솔
= 50바이트 남음
>4 415864865218675123
= 41바이트 남음
>4 1
= 32바이트 남음
>4 -9876543210321
= 23바이트 남음
>4 5678
= 14바이트 남음
>3 30320
= 9바이트 남음
>4 0
= 저장 공간이 가득 찼습니다.
6
415864865218675123LL 1LL -9876543210321LL 5678LL 30320I 0LL
D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\중간고사\64\Debug\중간고사.exe(프로세스 19744개)이(가) 종료되었습니다(
코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

그림 2

위 그림 2는 매우 큰 숫자와 음수가 입력되었을 때 잘 저장되는지 확인하기 위해 매우 큰 숫자와 음수를 입력한 것이다. 실행 결과 정상적으로 표시된다는 점을 알 수 있다.



## 4. 문제 2

세계 10개 도시를 여행하면서 부동산을 구입하고 통행료를 받는 게임을 구현한다. 플레이어는 2명이고 모두 컴퓨터가 자동으로 플레이한다. 각 플레이어는 5,000원으로 게임을 시작하며 주사위를 던져 나온 수만큼 움직인다. 이때 주사위는 1~6까지의 숫자만 적혀있다. 도착한 땅이 비어있고 이를 구매할 돈이 있으면 플레이어는 그 땅을 무조건 구입하고, 땅이 비어있으나 구매할 돈이 부족한 경우, 또는 이미 자신의 땅이면 아무런 행동을 하지 않고 턴을 종료한다. 다른 플레이어의 땅에 도착한 경우 통행료를 소유주에게 지불해야 하며 이때 지불할 돈이 부족하다면 해당 플레이어의 패배로 게임이 종료된다. 이미 구매한 도시를 처분할 수는 없고, 최대 30 턴 동안 진행하며, 땅 구매 비용은 300원, 통행료는 600원으로 땅 종류에 상관없이 일괄 적용된다. 명시된 것과 구조체를 활용해야 한다는 점 이외에는 제한사항이 없다.

## 5. 문제 2 해결 방안

먼저 도시와 플레이어의 정보를 저장하는 구조체를 각각 선언하였다. 플레이어의 정보를 저장하는 구조체에는 플레이어의 이름이 저장된 위치(포인터) 변수와 현재 그 플레이어가 보드 내에서 어디에 있는지를 저장할 도시 구조체 포인터, 그리고 현재 가지고 있는 돈의 양을 저장할 unsigned int 형 변수를 멤버 변수로 하였고, 도시 정보를 저장하는 구조체에는 똑같이 도시의 이름이 저장된 위치(포인터) 변수와 그 도시의 소유권자를 가리키는 플레이어 구조체 포인터, 그리고 구매가와 통행료를 저장할 unsigned int 형 변수 각각을 멤버 변수로 하였다. 모든 땅에서 구매가와 통행료가 일정한데 이를 각각 저장하는 것은, 이것이 달라질 때를 고려한 것이다. 실제 부루마블에서는 땅에 따라 구매가와 통행료가 달라지기 때문이다. 또 두 개의 구조체는 멤버로 다른 구조체의 포인터를 가지는 상황이 되는 것이다. 이후 그 구조체를 이용해 게임판(도시)과 플레이어들을 각 구조체의 배열로 선언했다. 이름이 저장된 위치(포인터)의 경우, 이름을 저번처럼 사용자의 입력을 받아 저장하는 것이 아니라 전부 미리 지정할 것이므로, 저번 과제에서 만들어 사용했던 sgets 함수를 이용하는 게 아니라 문자열 상수를 사용했다. 게임판을 저장하는 배열에서 도시의 소유권자를 가리키는 포인터는 아직 주인이 없다는 점을 나타내기 위해 모두 NULL로 초기화했고, 구매가와 통행료는 모두 각각 기본 설정값인 300과 600으로 초기화했다. 단 땅이 아닌 시작 위치의 경우 구매가와 통행료를 0으로 설정했다. 플레이어를 저장하는 배열에서도 이름은 똑같이 문자열 상수를 사용하고, 보드에서의 현재 위치는 모두 보드의 시작 위치를 가리키도록, 그리고 가지고 있는 돈은 모두 기본값인 5,000원으로 초기화하였다. 그리고 게임을 시작하기 전, 앞으로 랜덤 값을 활용할 테니 랜덤시드 초기화를 해 주었다. 게임이 시작되면 1부터 6까지의 무작위 수를 뽑아 현재 위치에서 그만큼 이동한다. 이때 현재 위치가 시작 지점에서 얼마나 떨어져 있는지는 현재 저장된 플레이어의 위치 포인터 값에서 보드 시작 위치의 포인터 값을 빼는 것으로 구할 수 있다(보드를 도시 구조체의 배열로 만들었기에 이를 활용할 수 있다). 이 값에 주사위

의 눈의 수(방금 뽑은 무작위 수)를 더한 뒤 이를 전체 보드의 크기로 나눠 주면 이번에 도착하는 칸이 시작 지점으로부터 얼마나 떨어져 있는 위치인지가 나온다. 여기에 시작 위치의 포인터 값을 더해주면 현재 해당 플레이어의 위치를 도시 구조체의 포인터로 나타낼 수 있다. 이렇게 업데이트한 플레이어의 위치를 저장한다( $p \rightarrow place = board + (((p \rightarrow place - board) + dice) \% board\_size);$ ). 이후 화면을 초기화한 뒤 보드 화면을 새로 화면에 표시한다. 그리고 이번에 나온 주사위 눈의 수를 화면에 표시한다. 그리고 도착한 도시의 상태에 따라 다음 과정을 진행한다.

도착한 곳이 시작 위치인 경우, 땅이 아니므로 도착한 곳이 시작 위치라는 것을 표시만 한 뒤 넘어간다. 도착한 땅의 주인이 없는 경우, 도착한 땅의 이름과 주인이 없다는 사실을 보여준 뒤, 현재 그 플레이어가 가지고 있는 돈이 그 도시의 구매가보다 적지 않으면 도시 구매를 진행한다. 도시를 살 돈이 없는 경우 구매할 수 없다는 말을 화면에 띄우고 넘어간다. 도착한 땅에 주인이 있으나 그게 내 땅인 경우 시작 위치에 도착했을 때와 비슷하게 도착한 땅의 이름과 땅 주인(턴을 진행 중인 플레이어 자신)의 이름을 띄운 뒤 넘어간다. 도착한 땅의 주인이 턴을 진행 중인 플레이어와 다른 경우, 땅의 이름과 땅 주인의 이름을 표시한 뒤, 턴을 진행 중인 플레이어가 통행료 지급 능력이 있는지 확인한다(현재 가지고 있는 돈이 통행료보다 크거나 같은지 확인한다). 지급 능력이 없는 경우 통행료를 지불할 수 없다는 정보를 화면에 표시한 뒤 파산자 정보를 저장하는 포인터에 현재 턴을 진행 중인 플레이어의 포인터를 저장하고 게임을 종료하며, 지급 능력이 있으면 통행료를 땅 주인에게 전달한 뒤 통행료를 전달했다는 정보와 통행료를 받은 땅 주인의 통행료 수취 후 잔고를 화면에 출력한다.

이후 턴을 진행 중인 플레이어의 잔고를 보여주고 사람이 화면을 볼 수 있도록 잠시 기다린 후, 바로 다음 사람의 턴이 되어 게임을 계속 진행한다. 모든 플레이어가 턴을 완료했으면 다시 첫 번째 사람의 턴으로 돌아간다. 이를 파산자가 나오거나 30번(지정된 최대 턴 수) 반복할 때까지 반복한다. 게임이 끝나고 파산자 정보가 저장되어 있으면 해당 파산자로 인해 게임이 종료되었다는 메시지를, 저장되어 있지 않고 NULL이 저장되어 있으면 지정된 턴 수가 지나 게임이 종료되었다는 메시지를 내보내며, 각 플레이어의 최종 잔고를 보여주며 게임을 종료한다.

보드를 화면에 출력할 때는 5개 칸씩 출력했는데, 매번 출력을 진행한 게 아니라 출력할 내용을 문자열에 저장해 두고 한꺼번에 출력했다. 먼저 5개 칸의 위쪽 경계를 +와 -를 이용해 표시했다. 이 아래 줄에는 보드 경계를 |로 표시하고 도시의 이름을 중앙정렬 해서 표시했다. 그 아래 줄에는 그 도시의 소유자 정보가 있는 경우 소유자 정보를 중앙정렬로 표시하고, 그렇지 않으면 그 공간만큼 공백으로 메꾸었다. 마지막으로 위쪽 경계 표시하듯 아래쪽 경계를 표시하고, 이렇게 저장된 내용을 한꺼번에 화면에 출력했다. 여기서 어떤 칸에 어떤 도시의 정보를 출력하는지는, 1행의 경우 5, 6, 7, 8, 9번 index에 저장된 도시의 정보를 순서대로 가져왔고(즉 이를 반복문으로 만드는 경우,  $5+i$ 로 표현할 수 있다) 2행의 경우 4, 3, 2, 1, 0번 index에 저장된 도시의 정보를 순서대로 가져왔다(즉 이를 5회 반복하는 반복문으로 만드는 경우 index 식은  $4-i$ 가 된다). (이때 1행인 경우  $5+i$ 를, 2행인 경우  $4-i$ 를 표현하기 위해,  $\#define boardpl\_cal ((board\_acr - k) - i * (k * 2 - 1))$ 를 사용했다.  $k$ 는 1행일 때는 0, 2행일 때는 1이고,  $board\_acr$ 은 5이므로,  $(board\_acr - k)$ 은 1행에서 5,

2행에서 4가 되고,  $(k * 2 - 1)$ 은 1행에서 -1, 2행에서 1이 되어  $i$ 를 더하는지 빼는지를 조절할 수 있다.) 실행 예시에 플레이어의 위치를 표시하는 곳이 없는 관계로 플레이어의 위치를 표시하지는 않았다. 1행과 2행 사이에는 화살표를 표시해야 하는데, 화살표의 머리로 슬래시와 역슬래시를 이용했으며, 꼬리는 '|'을 이용했다. 다만 슬래시는 '/'으로 잘 표기되나, 역슬래시가 한국 화폐 단위로 표시되는 경우 화살표가 정상적으로 표시되지 않을 수 있다. 이렇게 역슬래시가 정상적으로 보이지 않는 경우까지 고려하지는 않았다. 화살표 출력의 경우, 1열 위치에 중앙정렬로 화살표의 머리를 먼저 만들어 주었다. 5열 위치에 화살표의 꼬리를 만들어 주고 다시 다음 줄에 1열 화살표의 꼬리를 만들어 주었으며, 앞에서 한꺼번에 출력하기 위해 선언한 문자열(str)이 감당할 수 있는 만큼(꼬리 3줄) 꼬리를 길게 만들어 주었다. 마지막으로 5열 화살표의 머리를 만들어 주고 이를 한꺼번에 화면에 출력하여 화살표를 구현했다.

이렇게 출력할 것을 따로 저장해 두었다 한꺼번에 출력하는 방식을 main 함수에서 사용하지 않은 이유는, 문자열 길이 계산이 어려울 것으로 예상되었기 때문이다.

이를 이용하여 문제 2를 해결한 결과는 다음과 같다.

## 6. 문제 2의 결과

문제 2의 코드는 숫자를 입력할 때마다 화면에 나타난 모든 글자를 지우므로, 화면을 지우기 전 출력 결과를 하나하나 캡처했다.

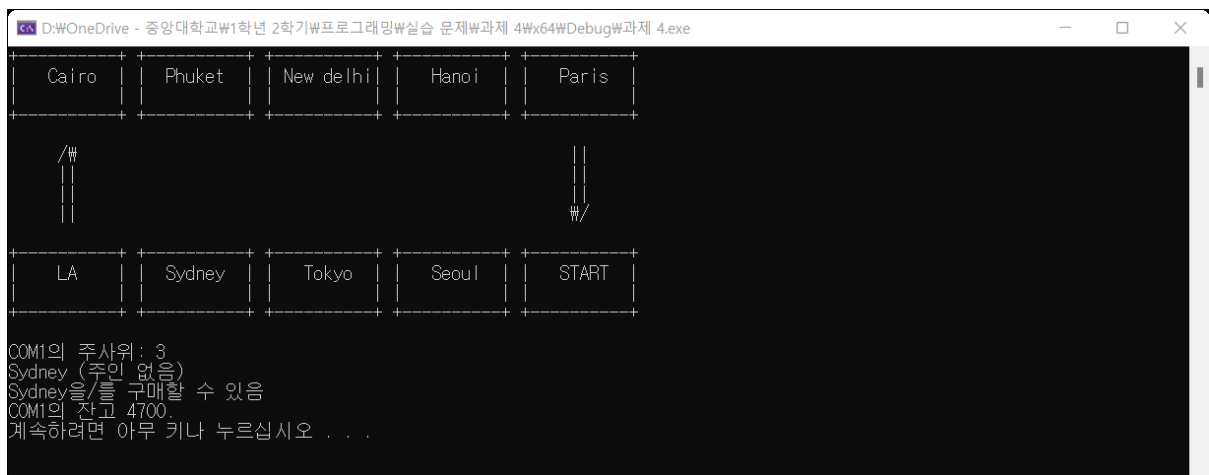


그림 4

이 실행 캡처본은 역슬래시가 한국 화폐 단위로 보이는 상황이다. '/W'는  $\Delta$ , 'W/'는  $\nabla$ 로 이해하여야 한다. 이점 착오 없기 바란다. 또한 보드에 플레이어별 현재 위치가 별도로 보이지 않으므로 이전 플레이어의 위치는 2번째 전 캡처본을 통해 확인할 수 있다.

그림 4에서 땅 구매시의 화면 출력을 확인할 수 있다.

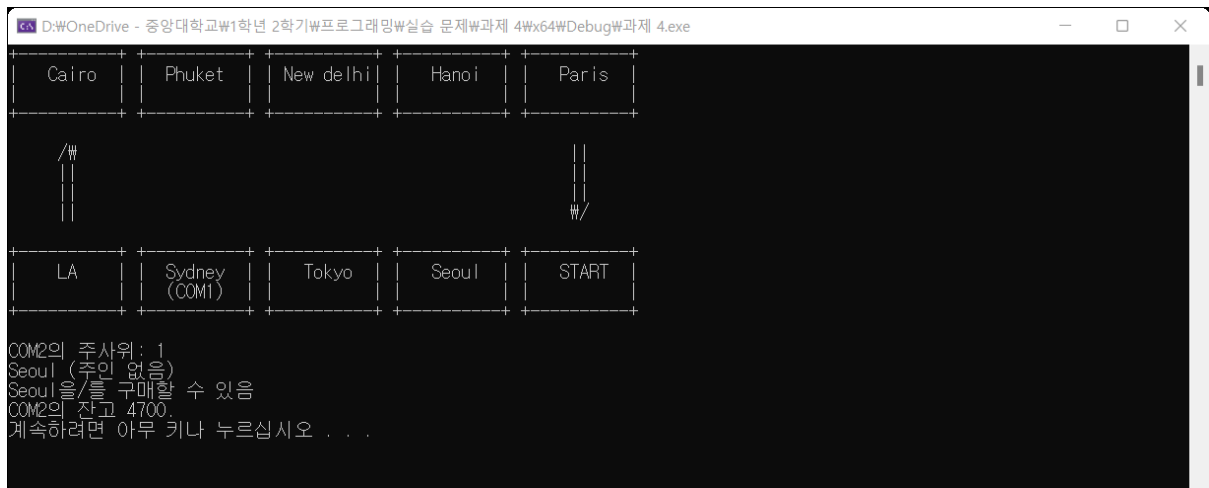


그림 5

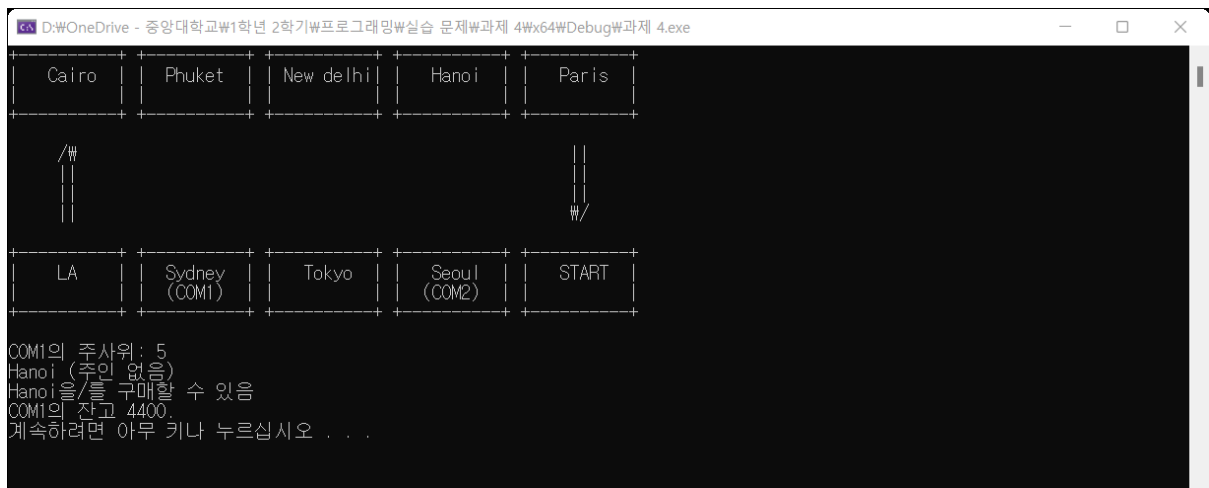


그림 6

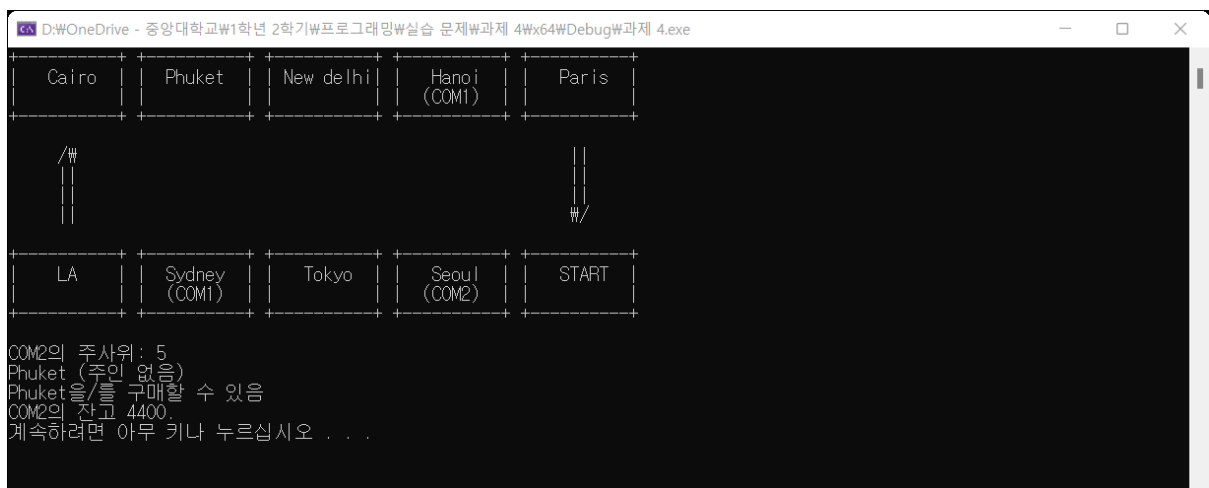


그림 7



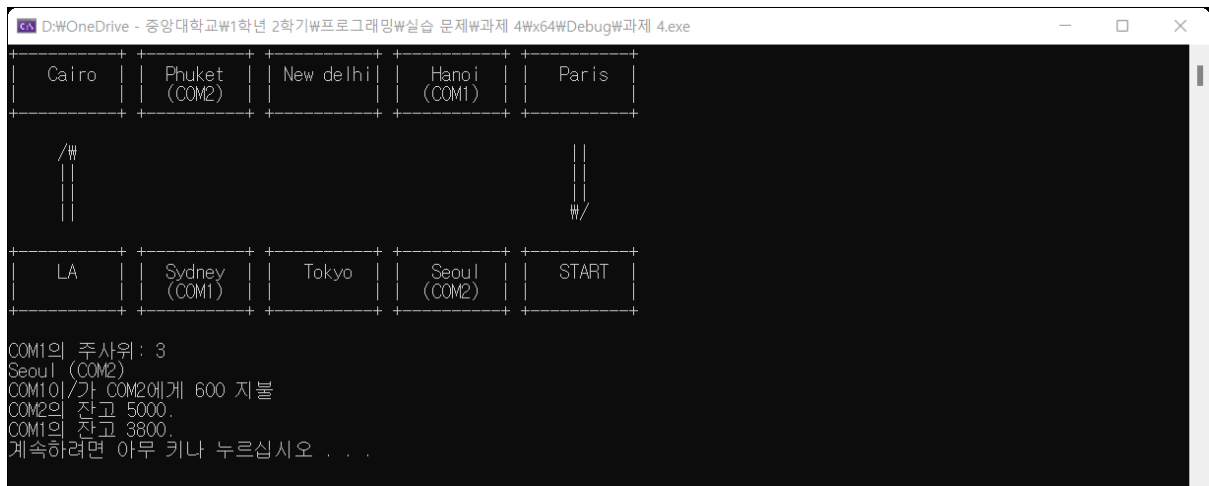


그림 8

그림 8에서 첫 번째 통행료 지불이 발생했다. 그림 8에서 통행료 지불 시의 화면 출력이 어떠한지를 확인할 수 있다.

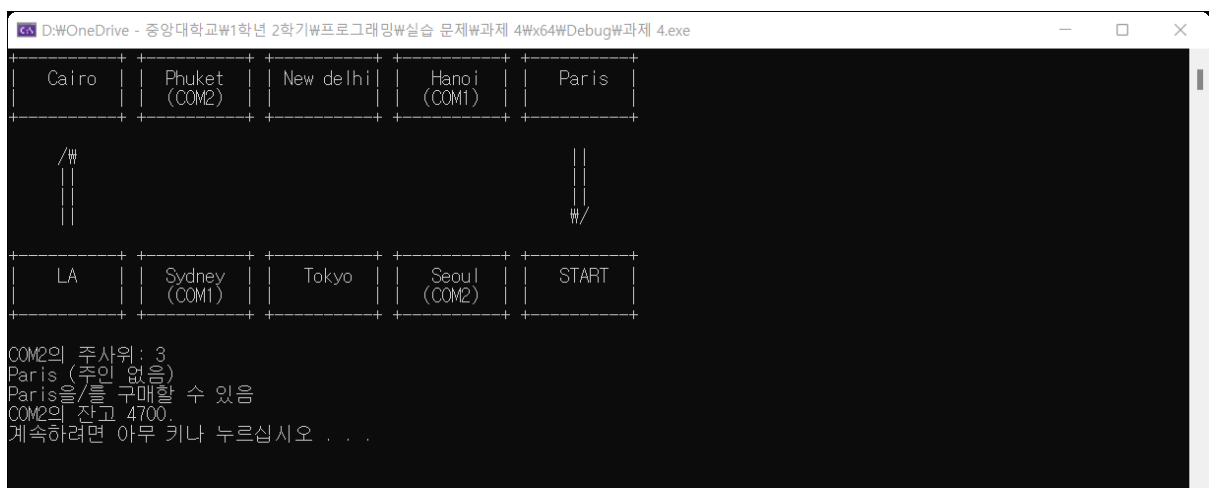


그림 9

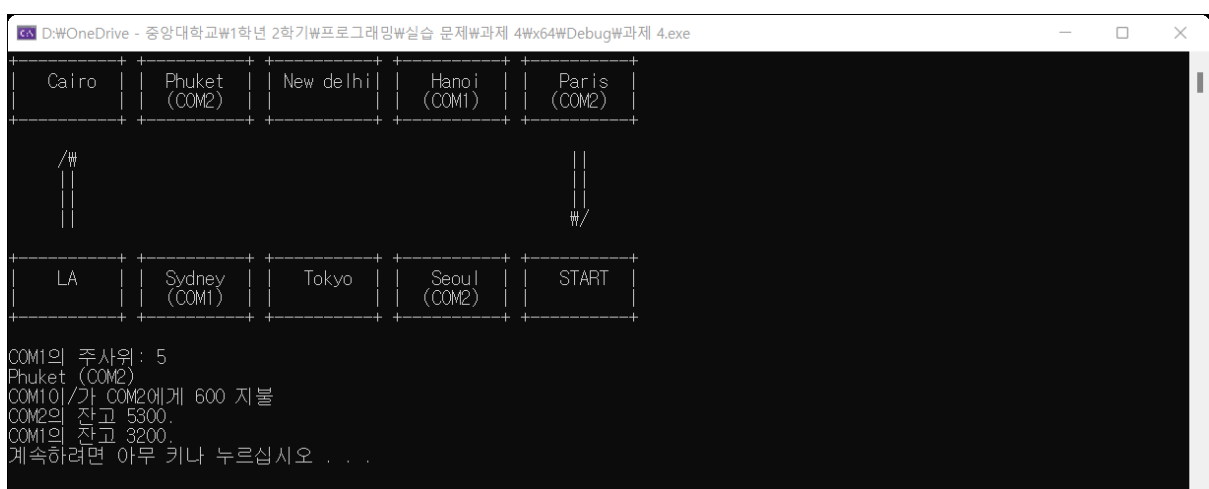


그림 10

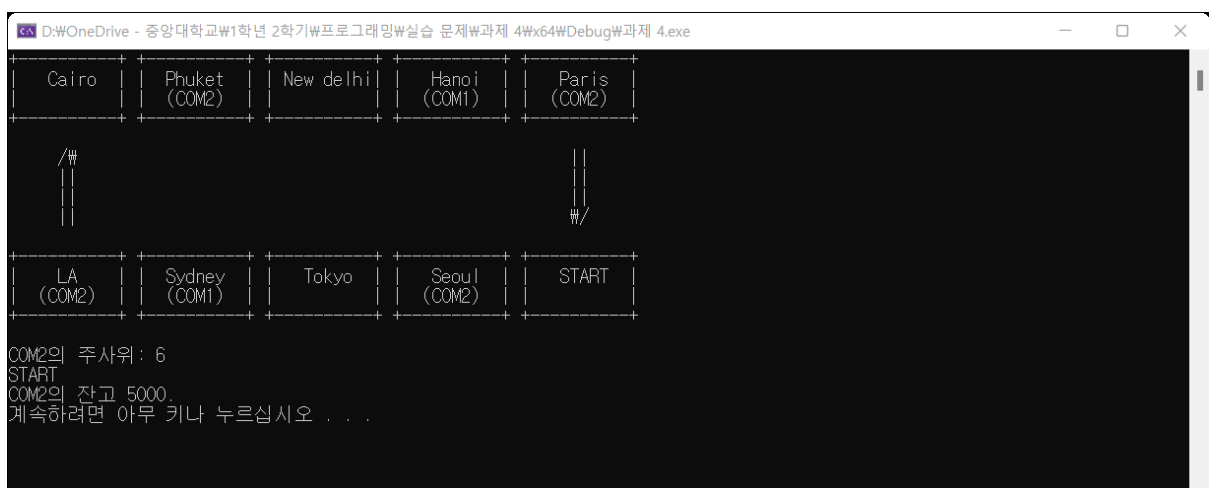
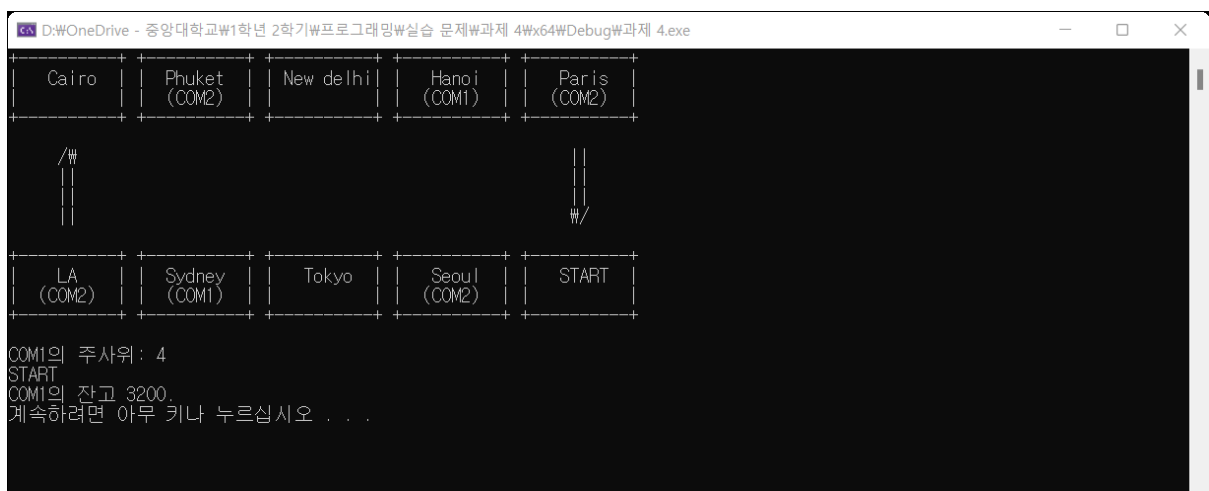
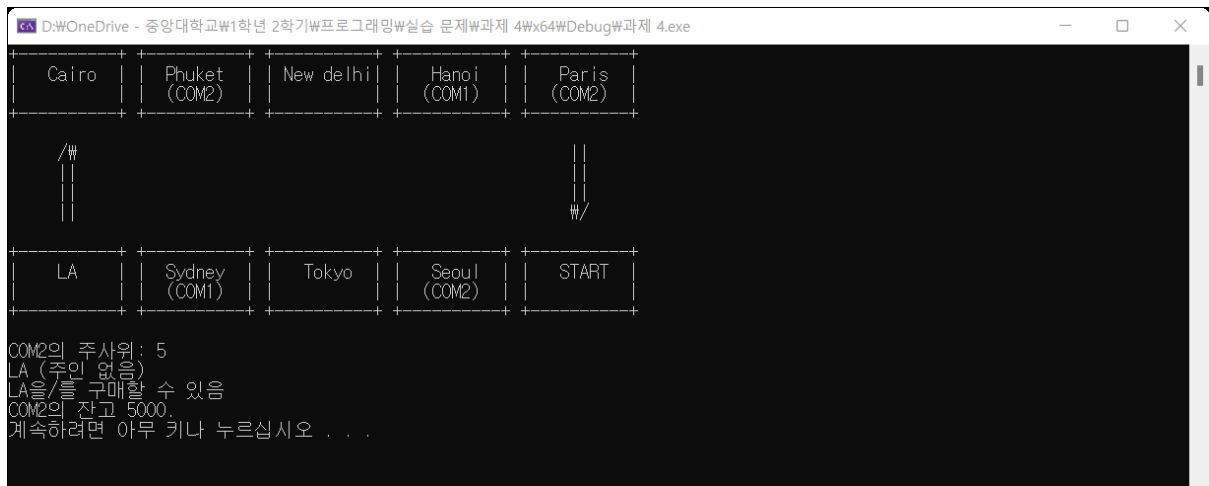


그림 13에서 START는 땅이 아니므로, 자기 땅인 것처럼 도착해도 아무런 일이 일어나지 않는다는 점을 확인할 수 있다.

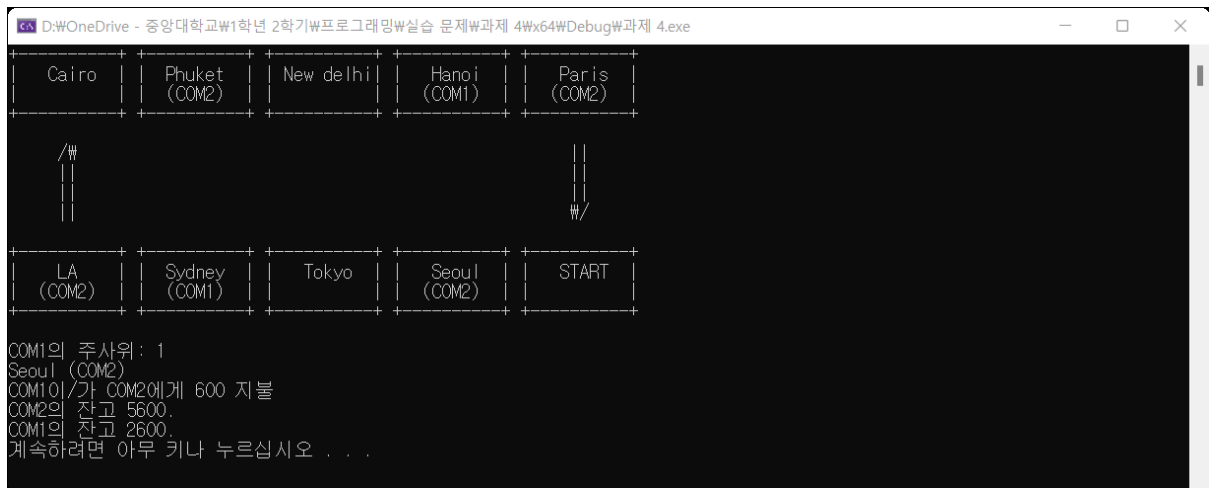


그림 14

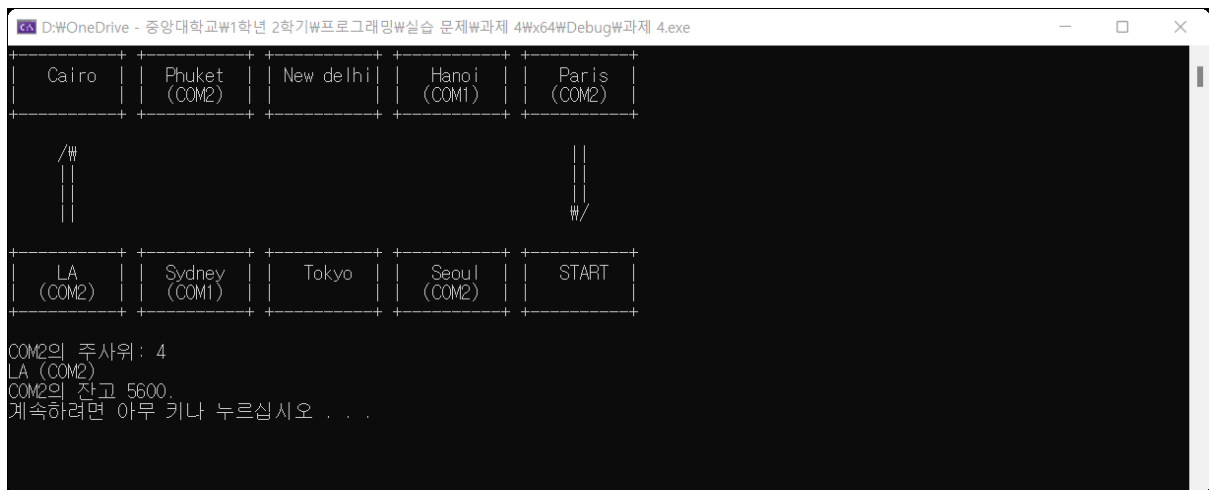


그림 15

그림 15에서 자신의 땅에 도착해도 아무런 일이 발생하지 않는다는 점을 확인할 수 있다.

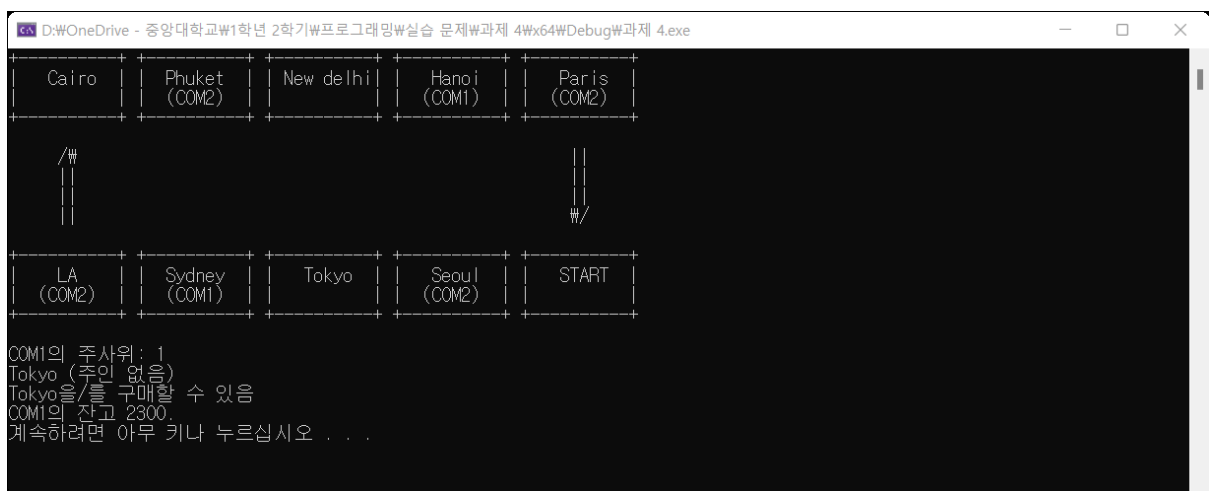


그림 16

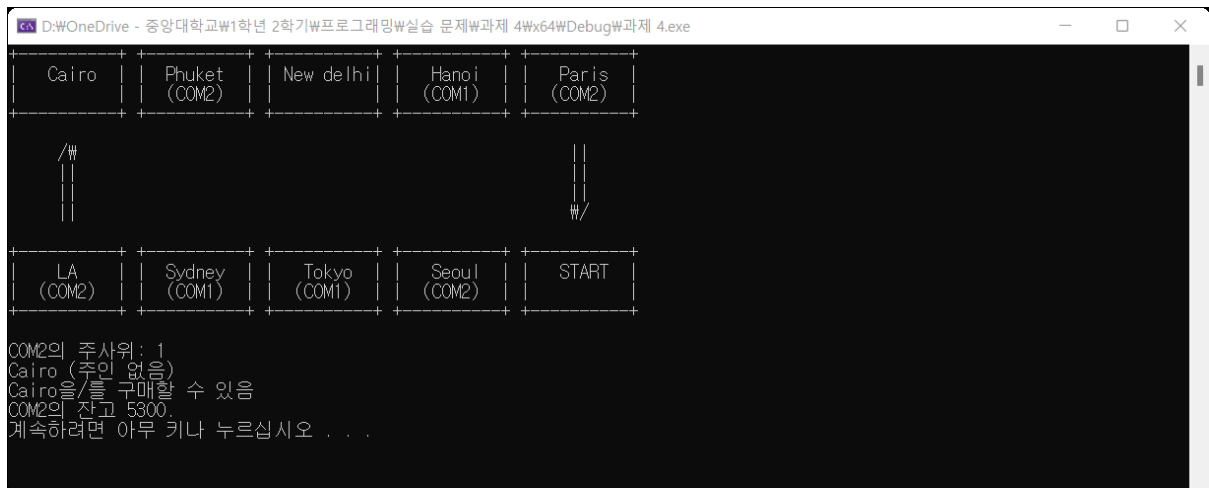


그림 17

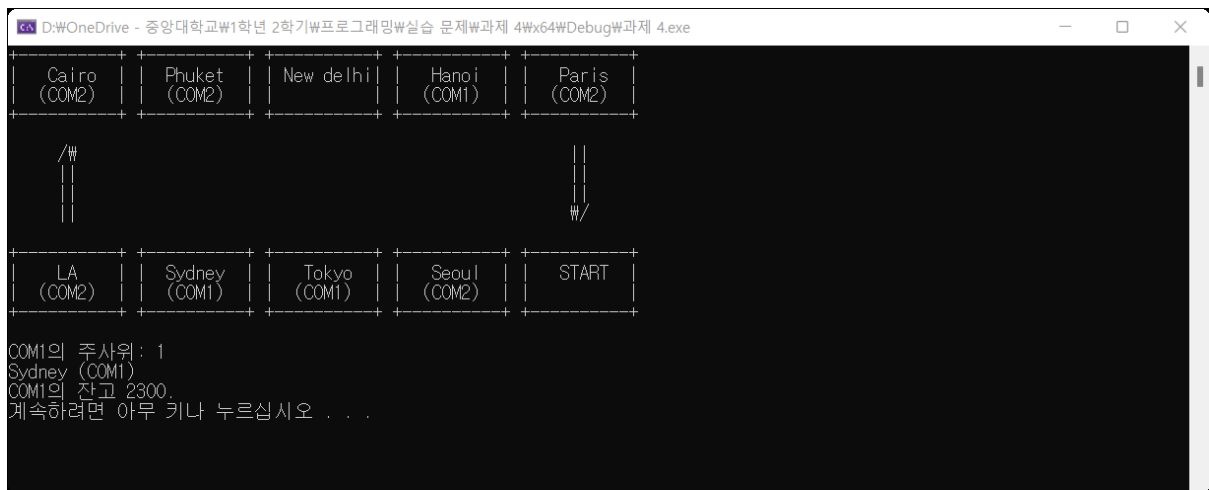


그림 18

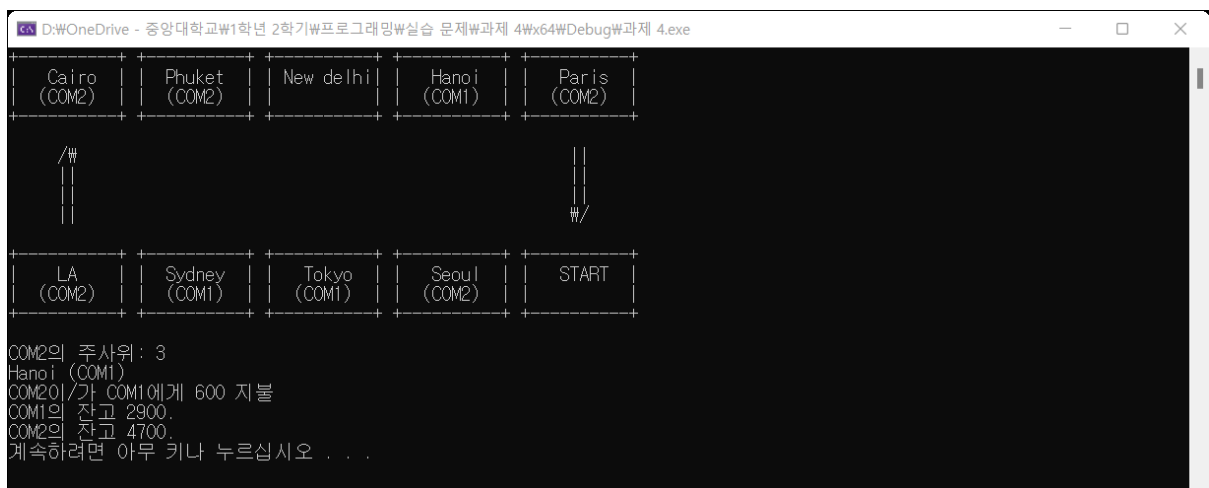


그림 19

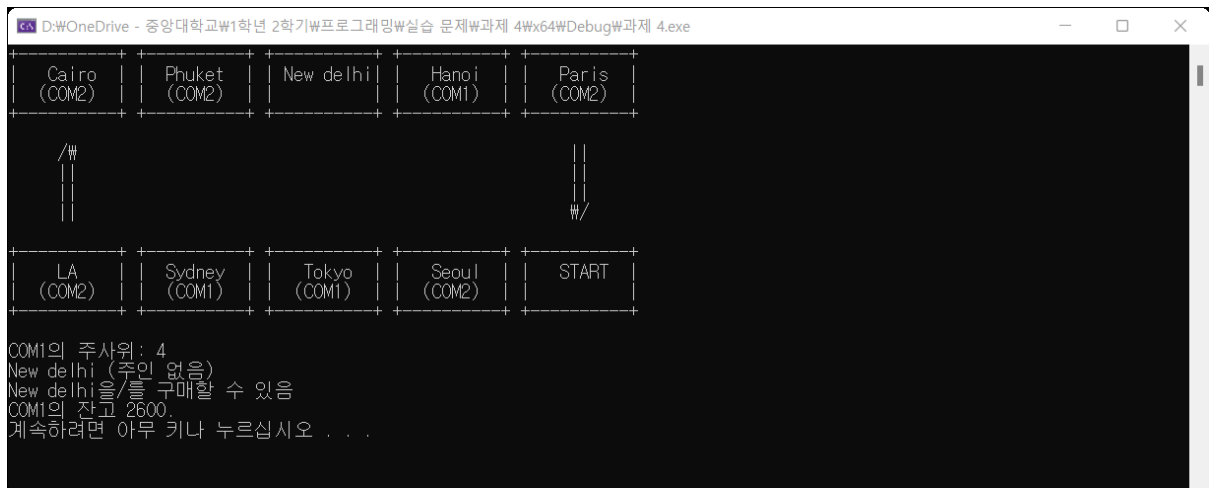


그림 20

그림 20의 결과로 모든 땅이 주인을 찾아가게 되었다. 이제 게임 종료시까지 추가 땅 구매는 일어나지 않는다.

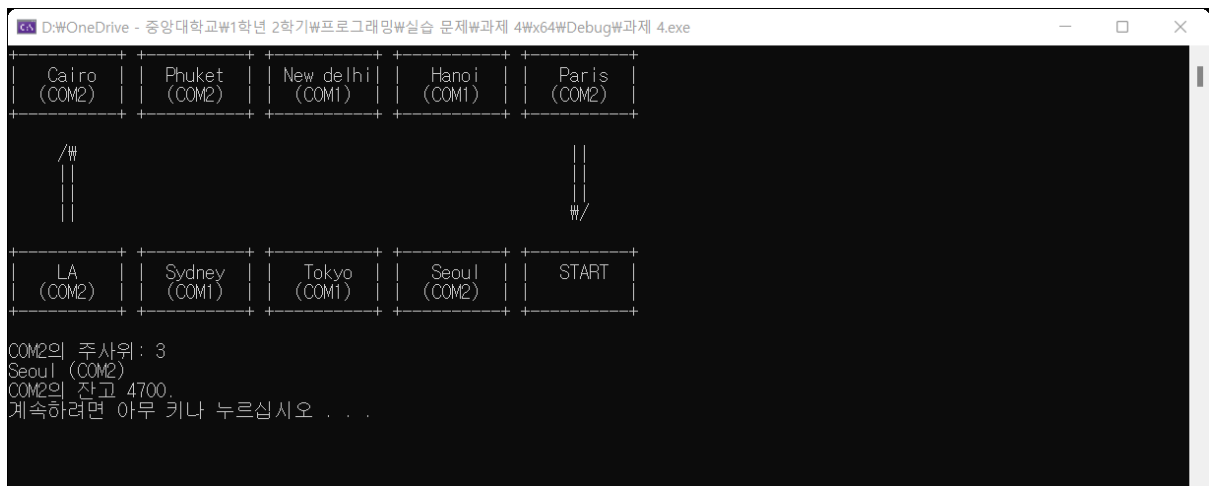


그림 21

그림 21에서 보드에 모든 땅이 주인을 찾아가게 된 것이 적용되었음을 알 수 있다.

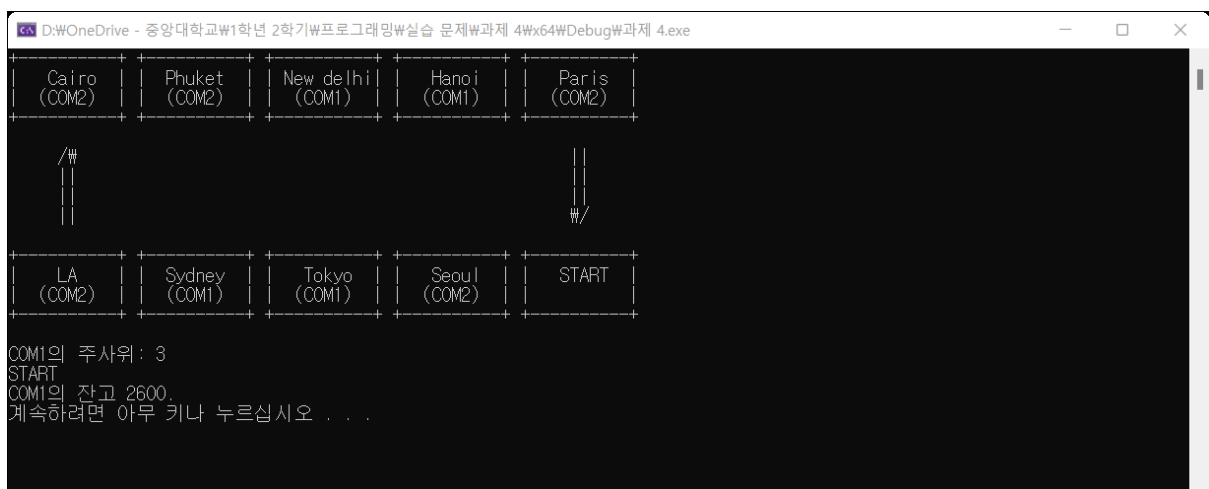


그림 22

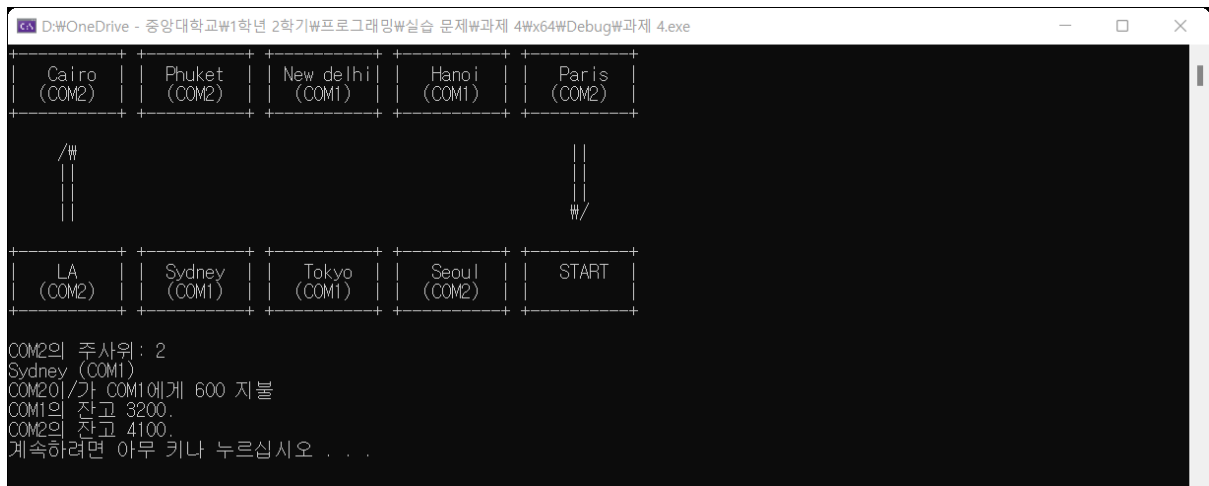


그림 23

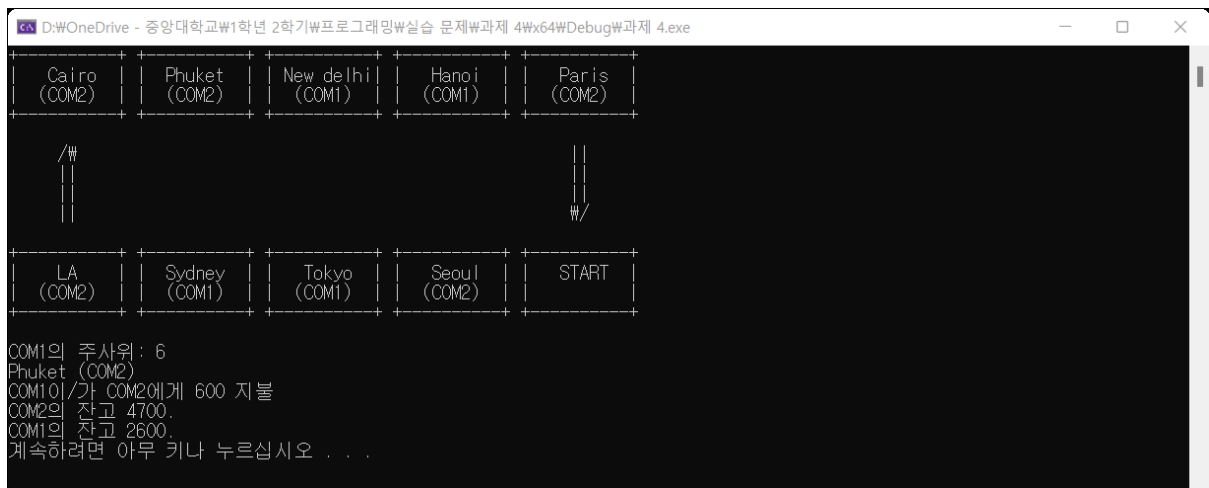


그림 24

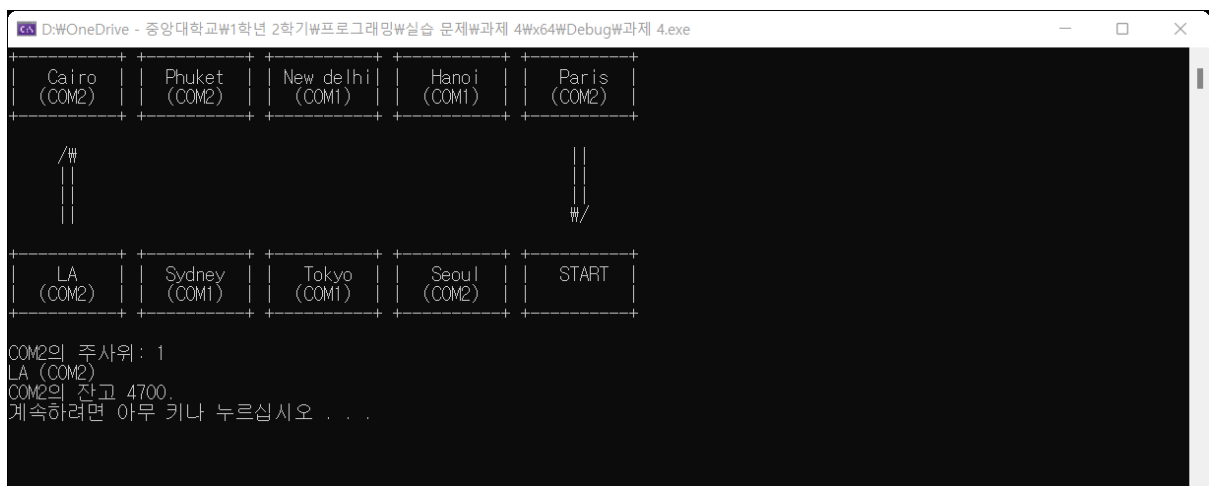


그림 25



그림 26

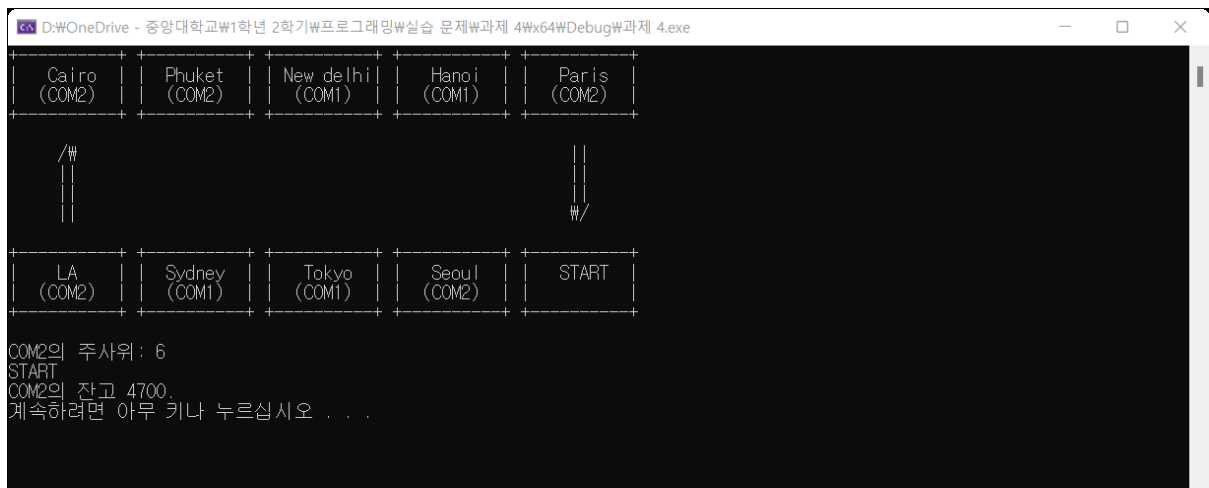


그림 27

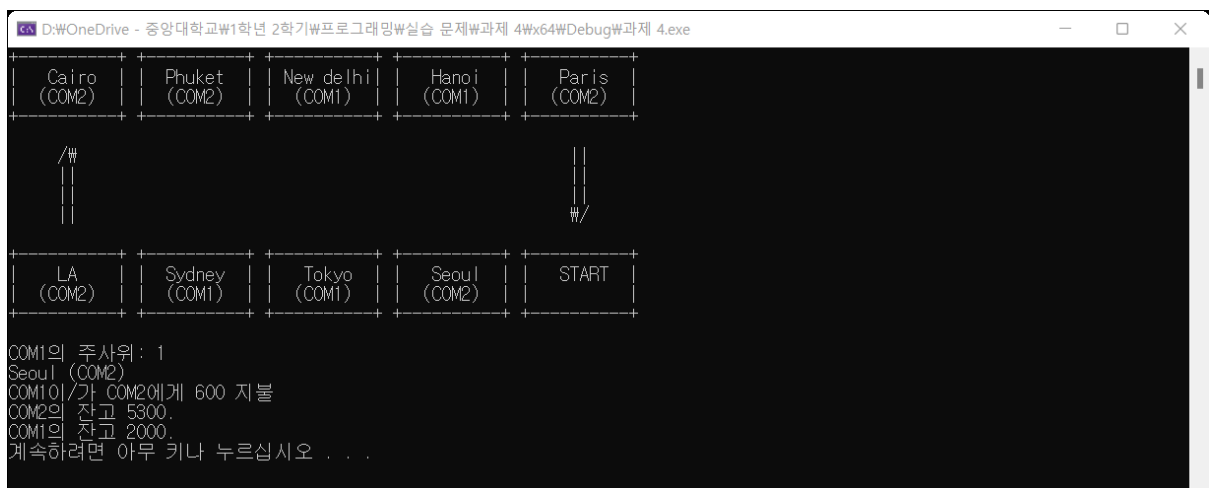


그림 28

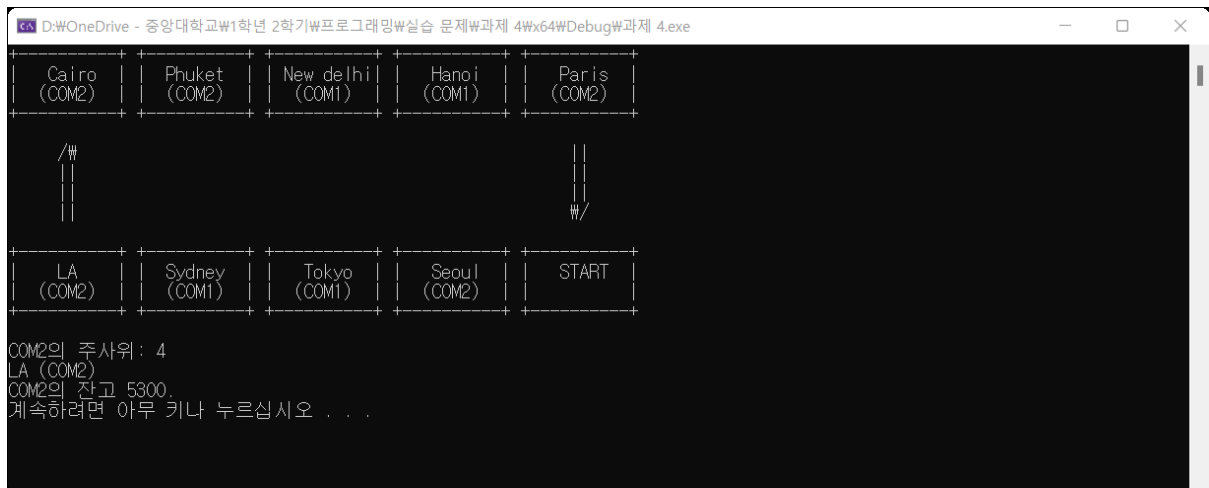


그림 29

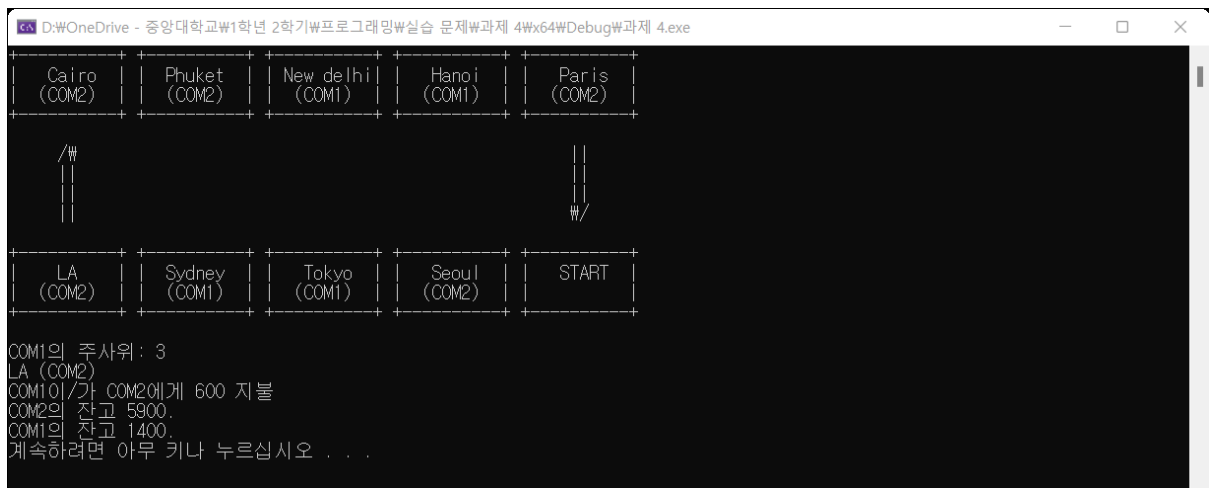


그림 30

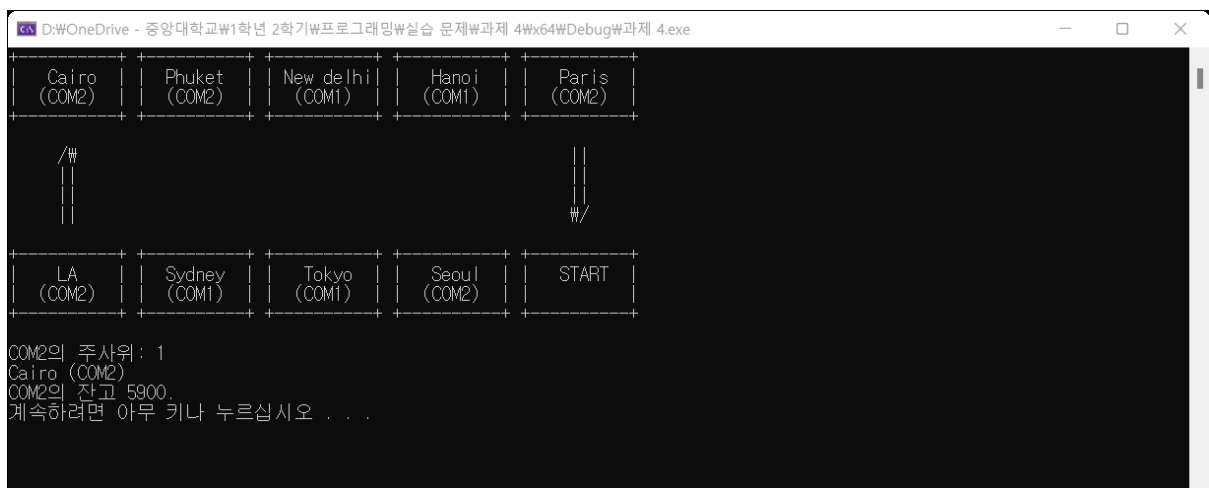


그림 31





그림 32

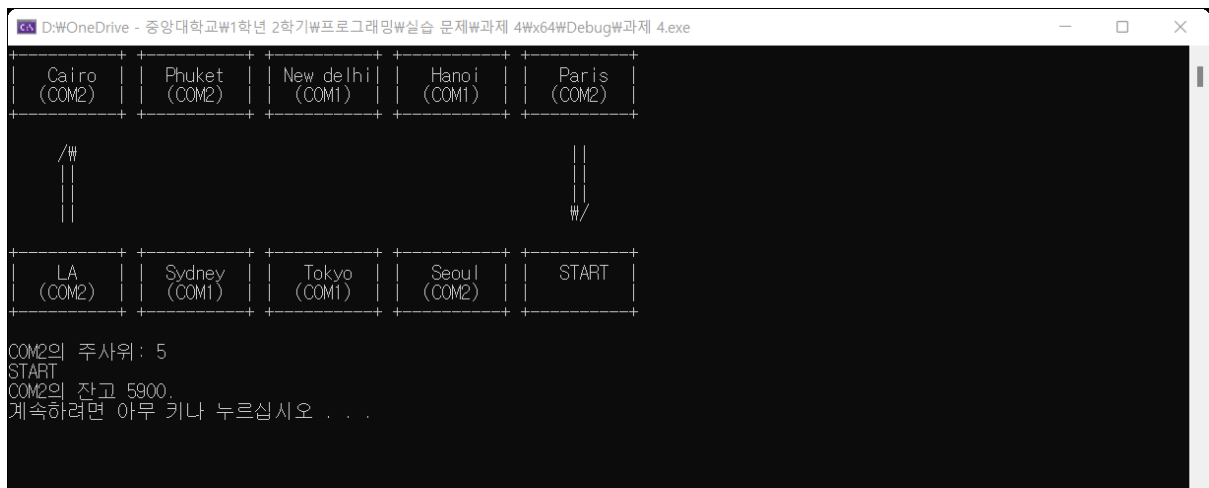


그림 33

그림 33까지 게임이 진행된 결과 설정된 턴의 절반인 15턴이 경과하였다. (그림 30개 / 2 플레이어 = 15턴)

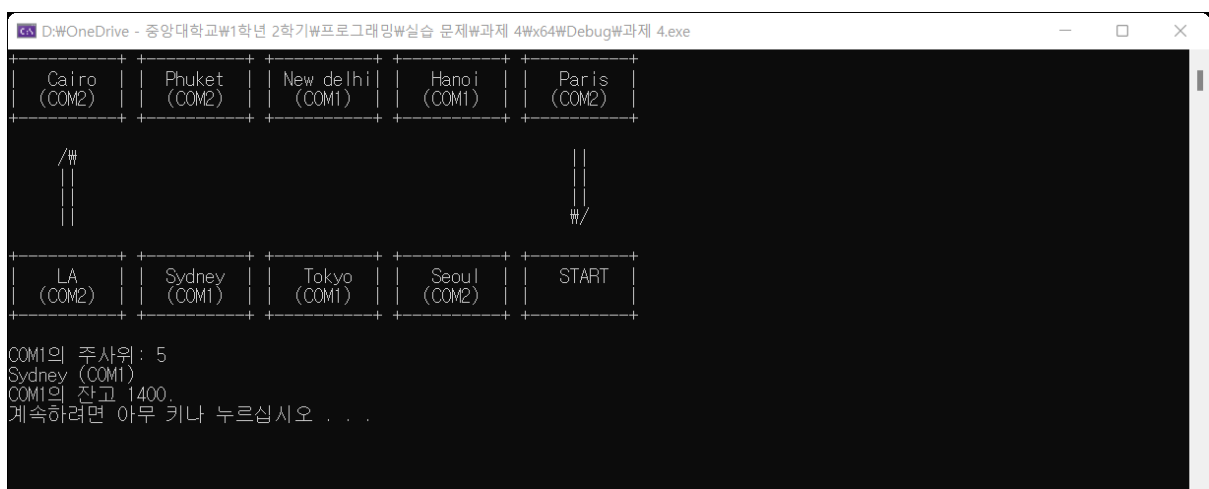


그림 34



그림 35

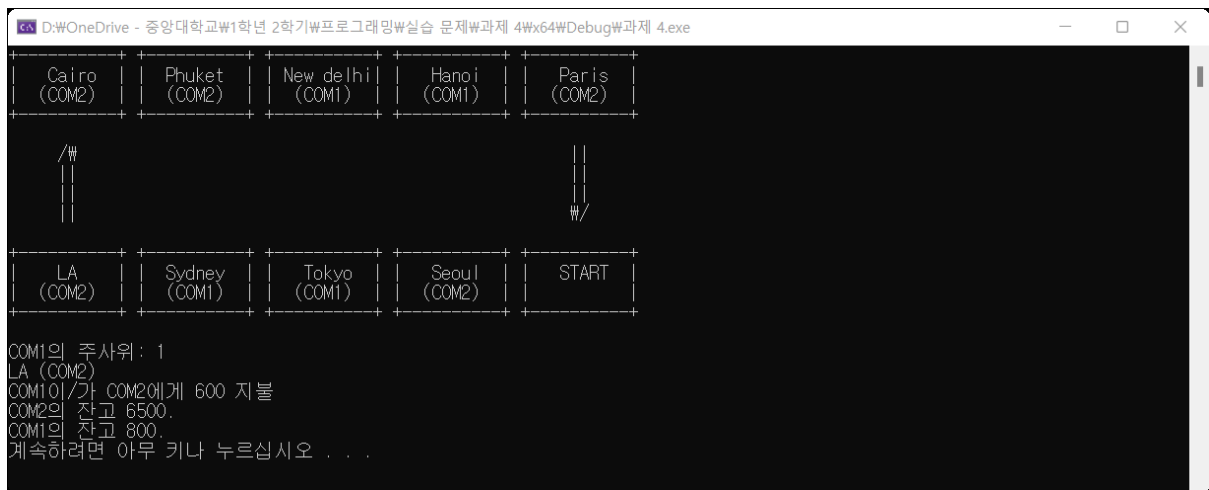


그림 36

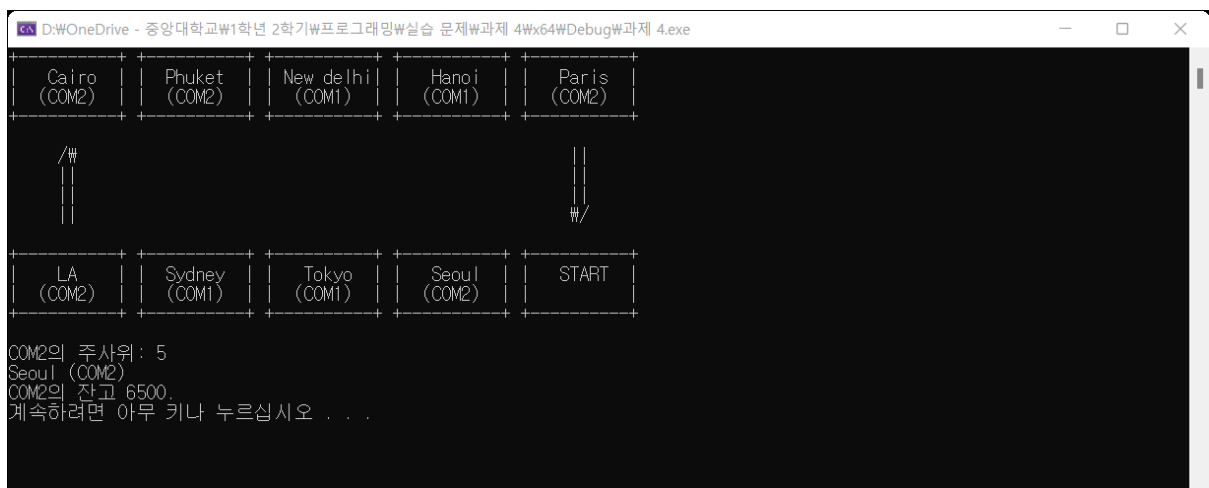


그림 37



그림 38

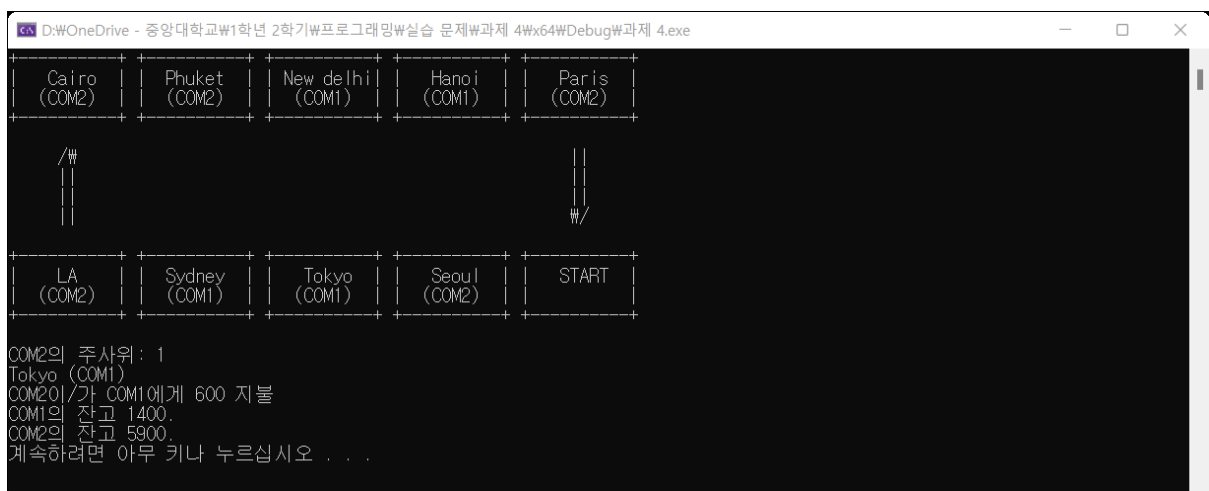


그림 39

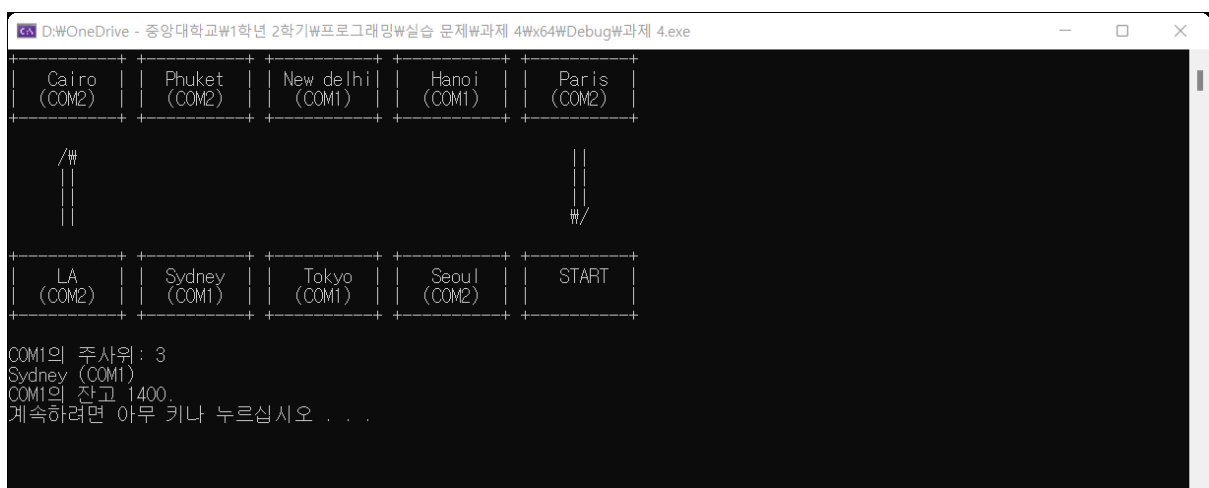


그림 40

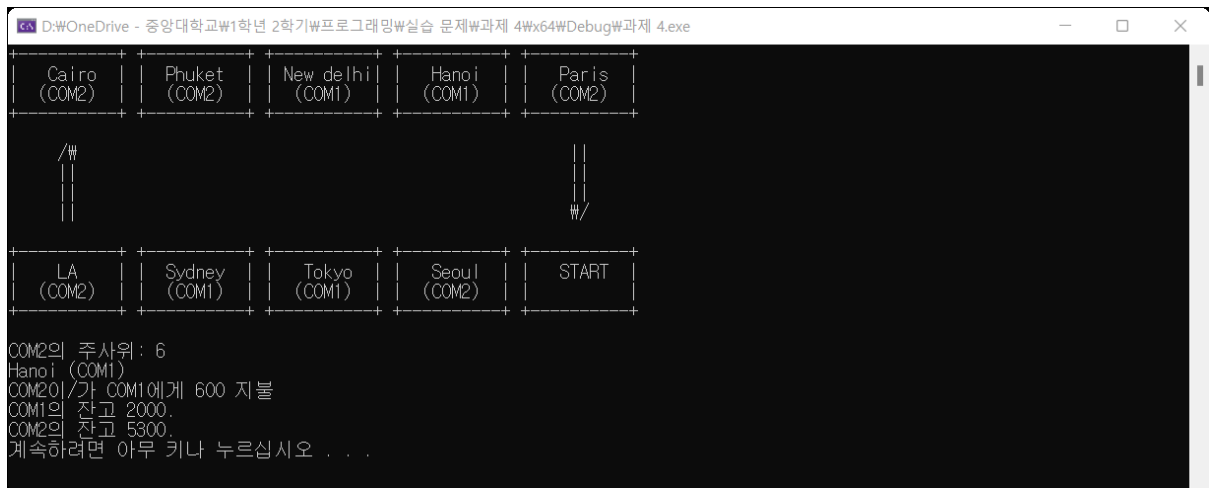


그림 41

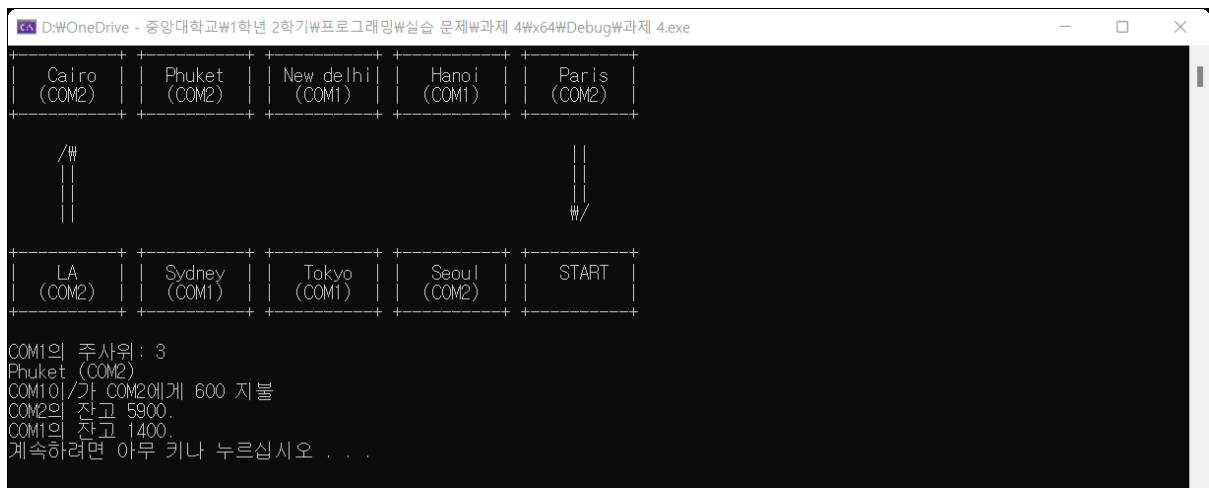


그림 42

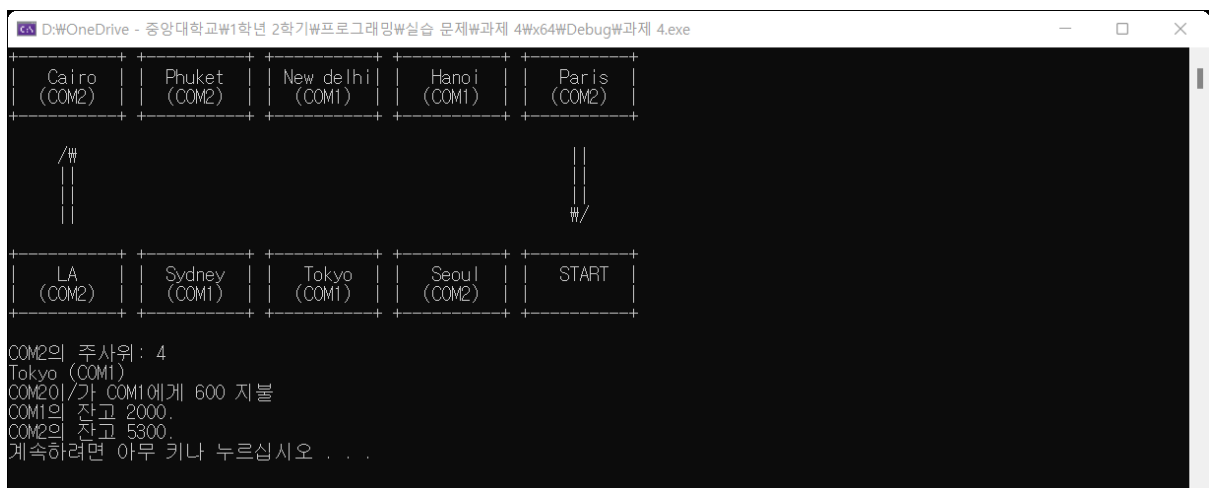


그림 43

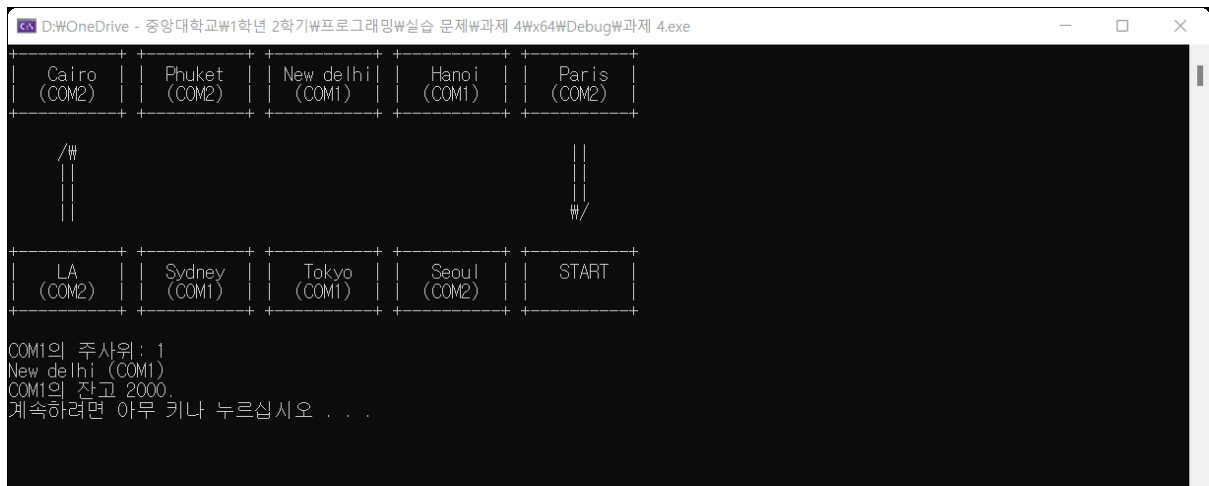


그림 44

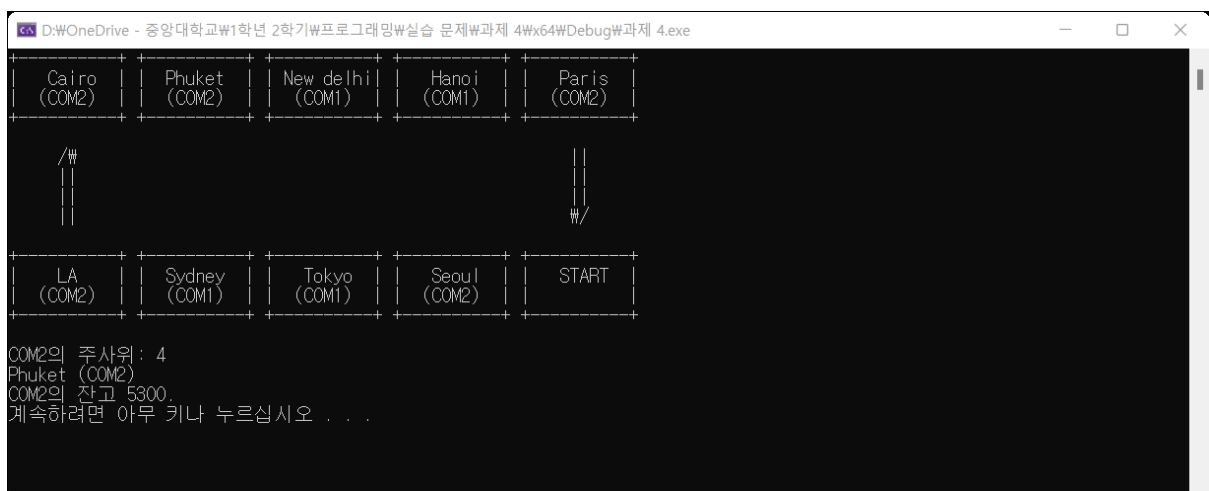


그림 45

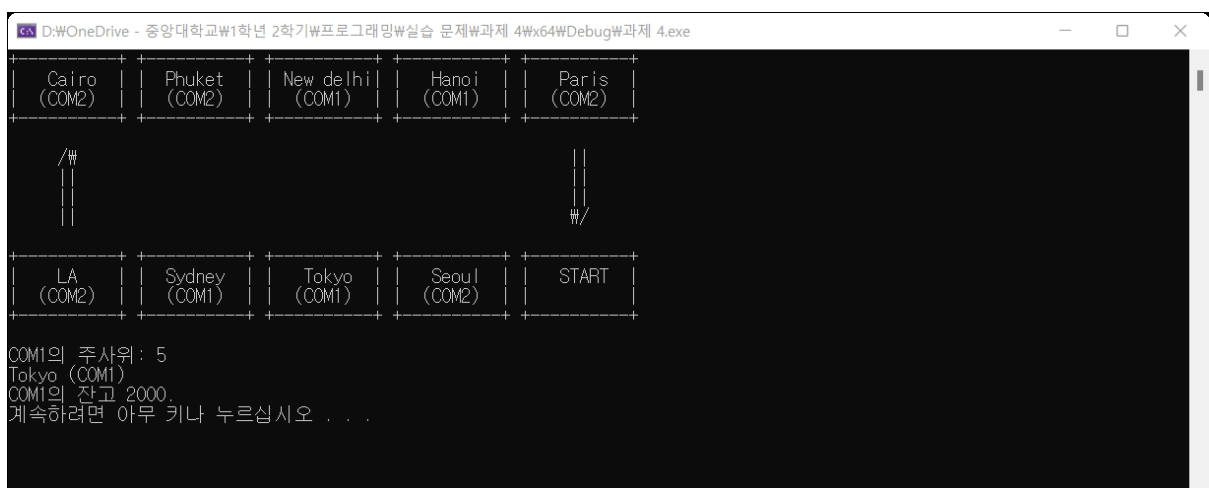


그림 46

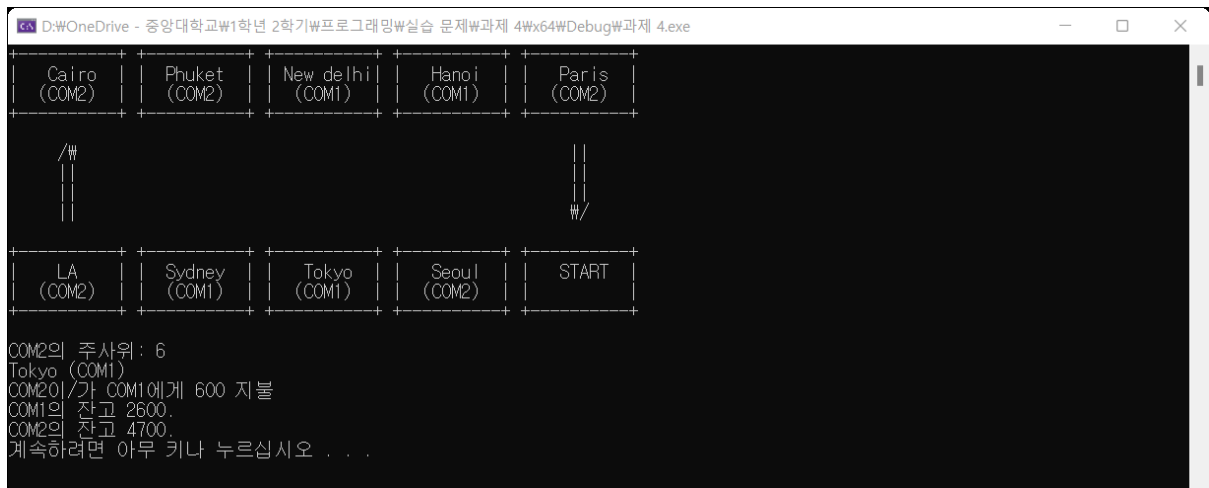


그림 47

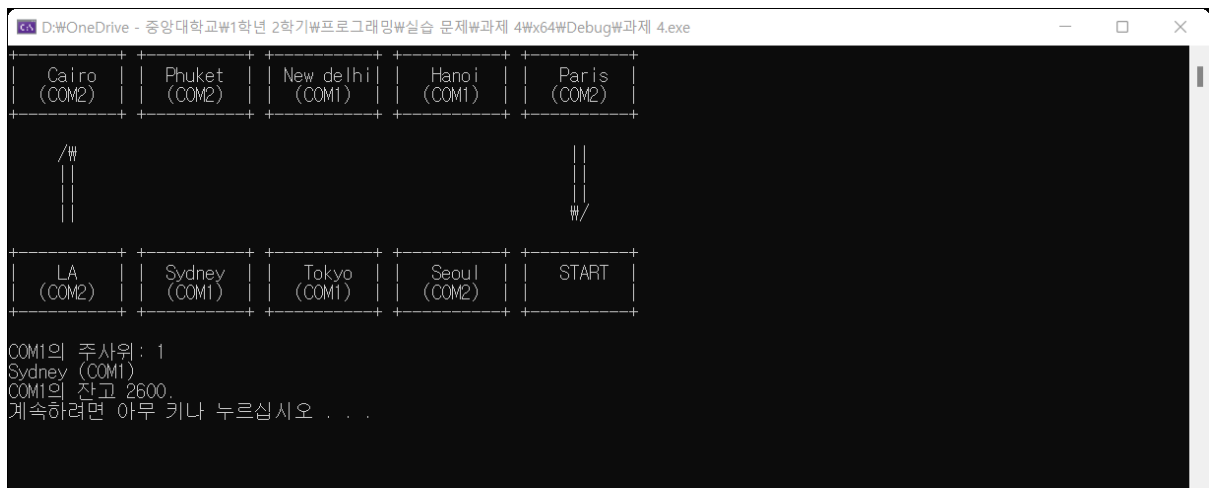


그림 48

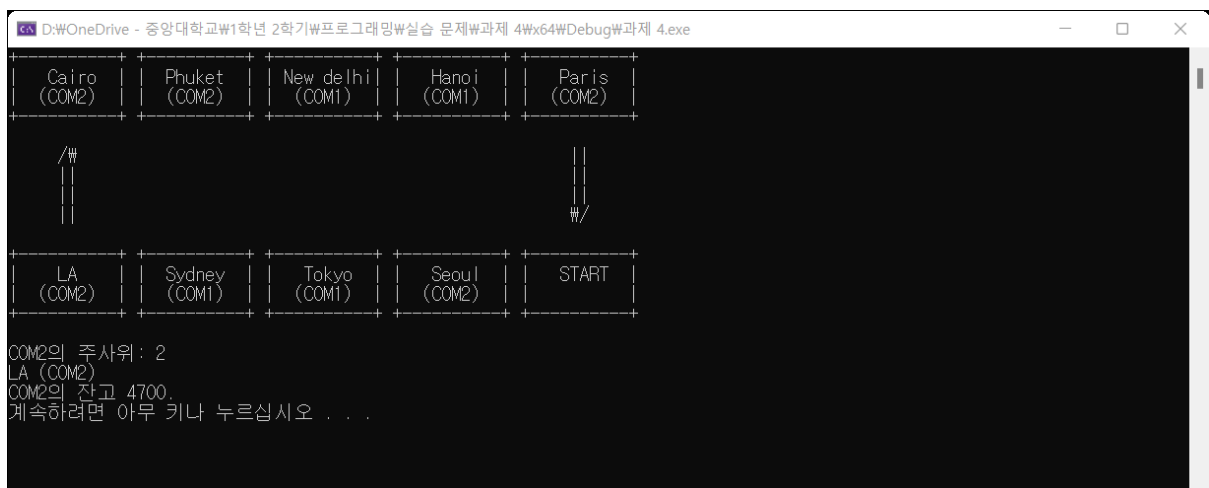


그림 49

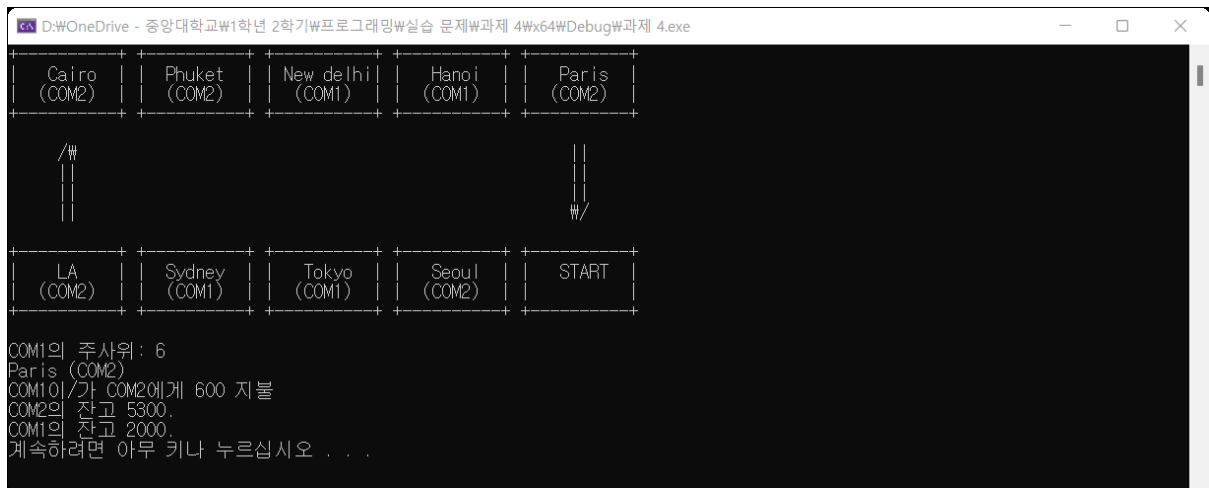


그림 50

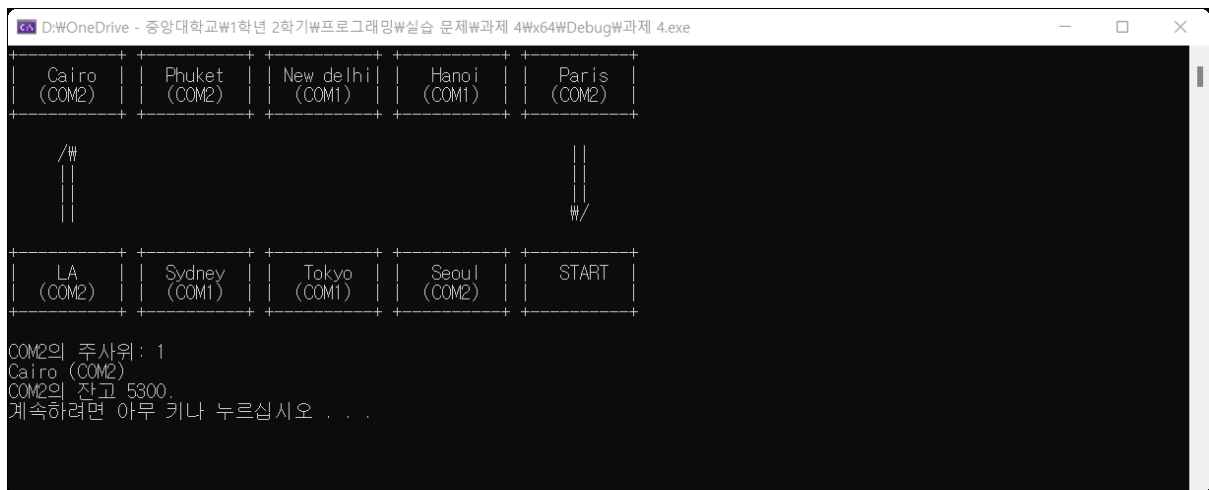


그림 51

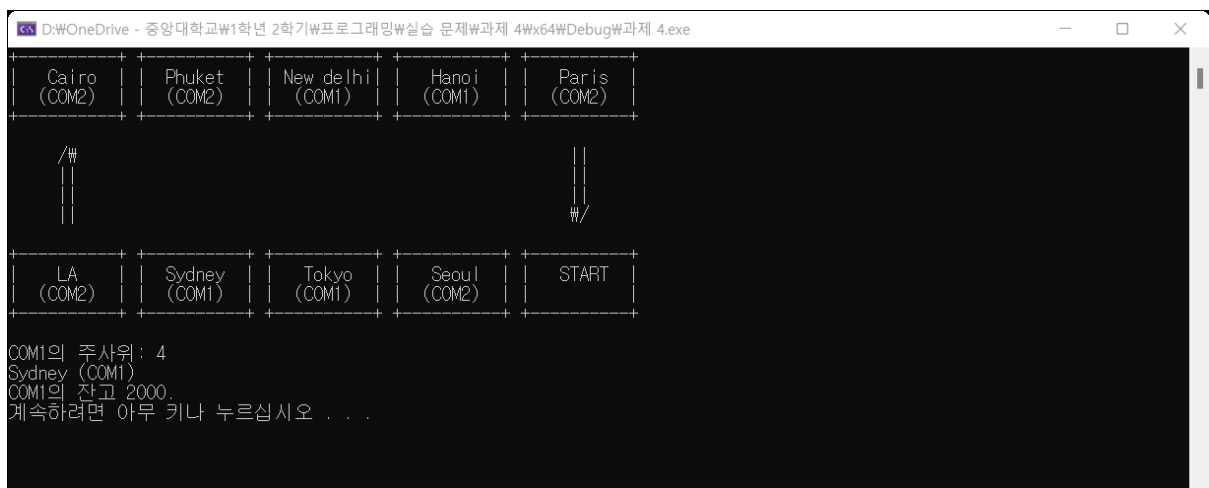


그림 52

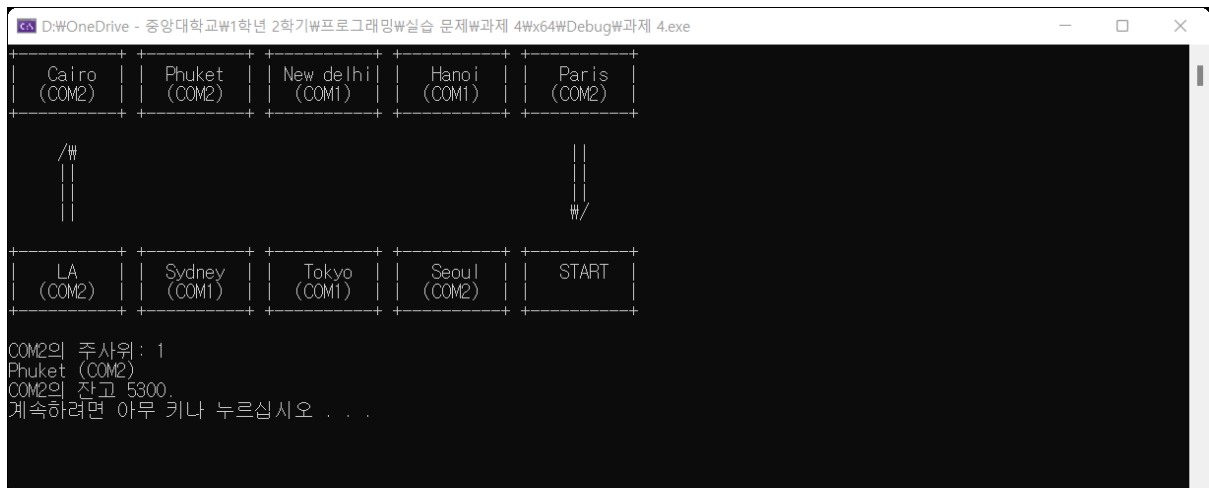


그림 53

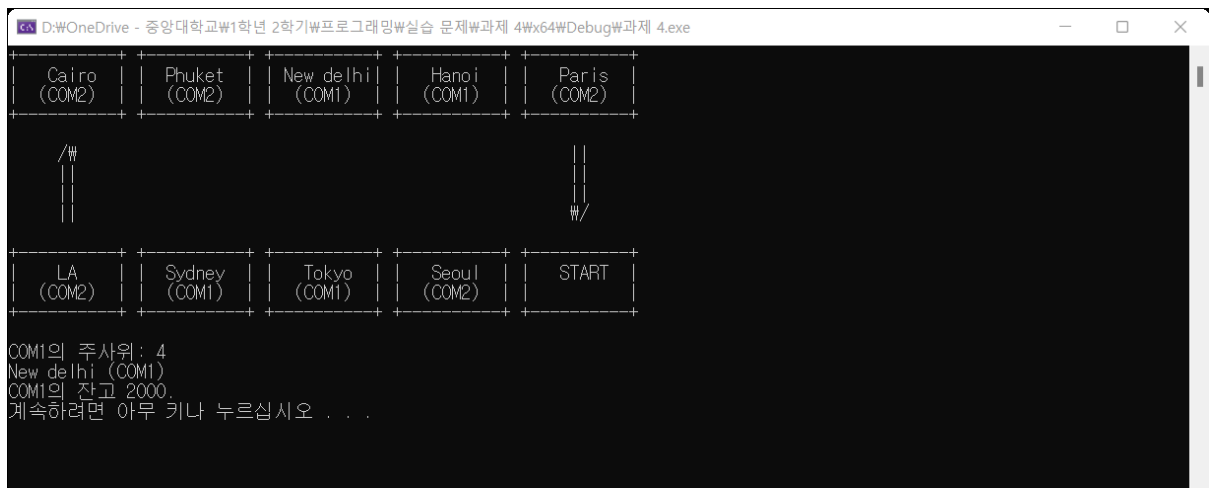


그림 54

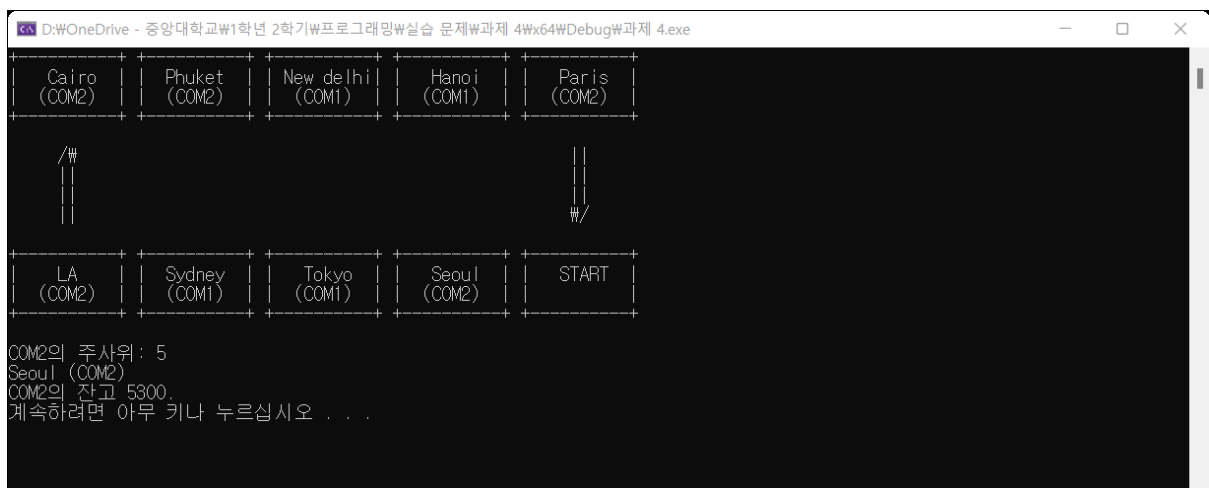


그림 55



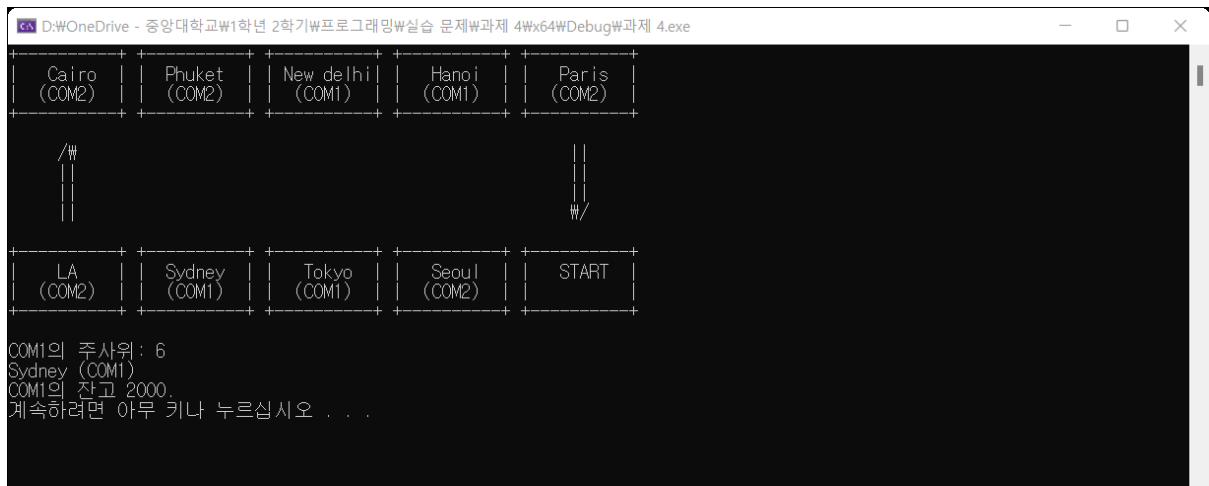


그림 56

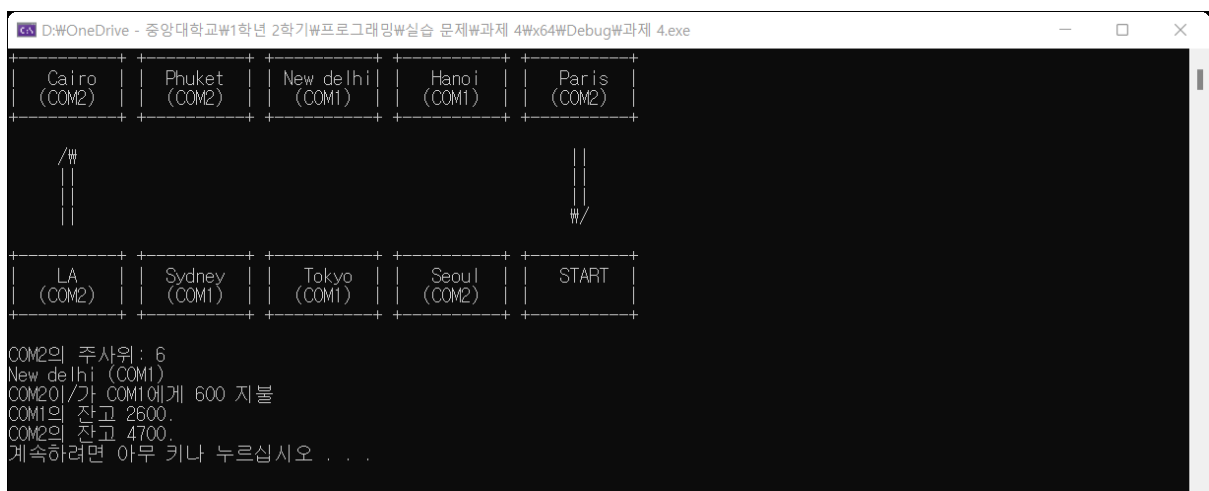


그림 57

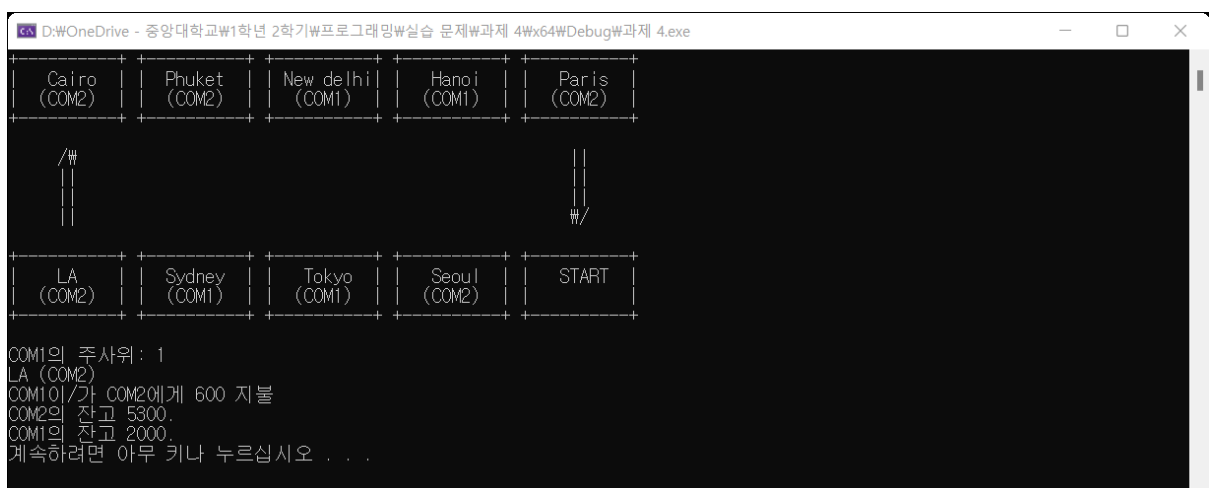


그림 58



그림 59

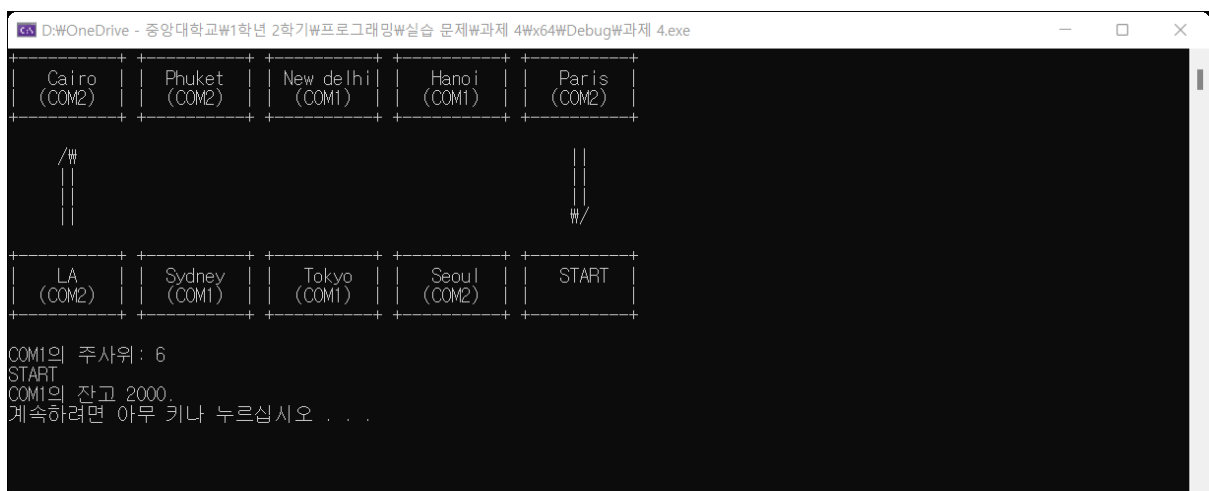


그림 60

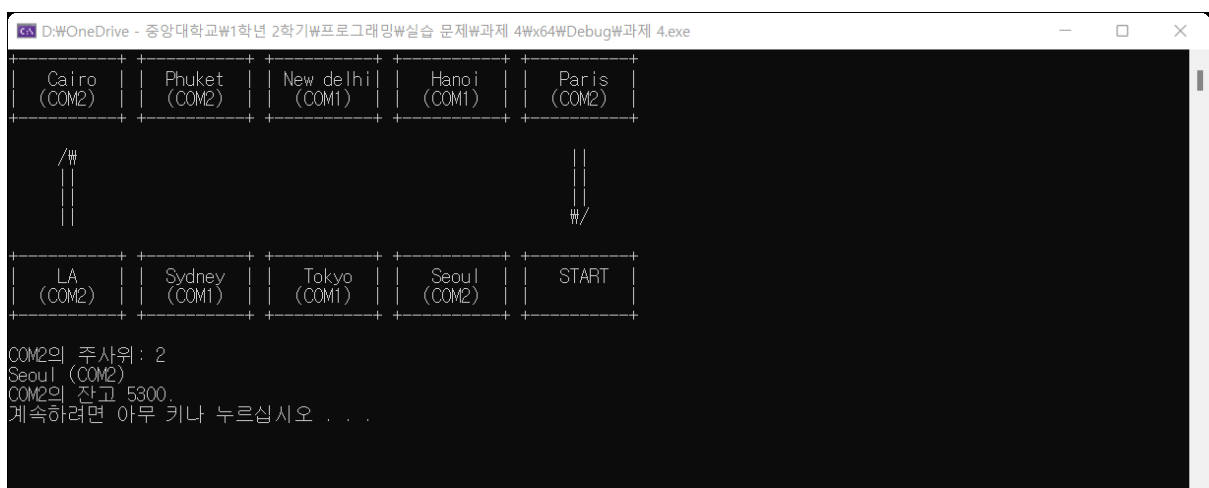


그림 61

그림 61의 화면까지 29턴이 지나 이제 각 플레이어의 마지막 턴이다. 물론 화면에 턴 숫자가 표시되지는 않으니 별도로 세는 게 아닌 이상 이를 알 수는 없다.

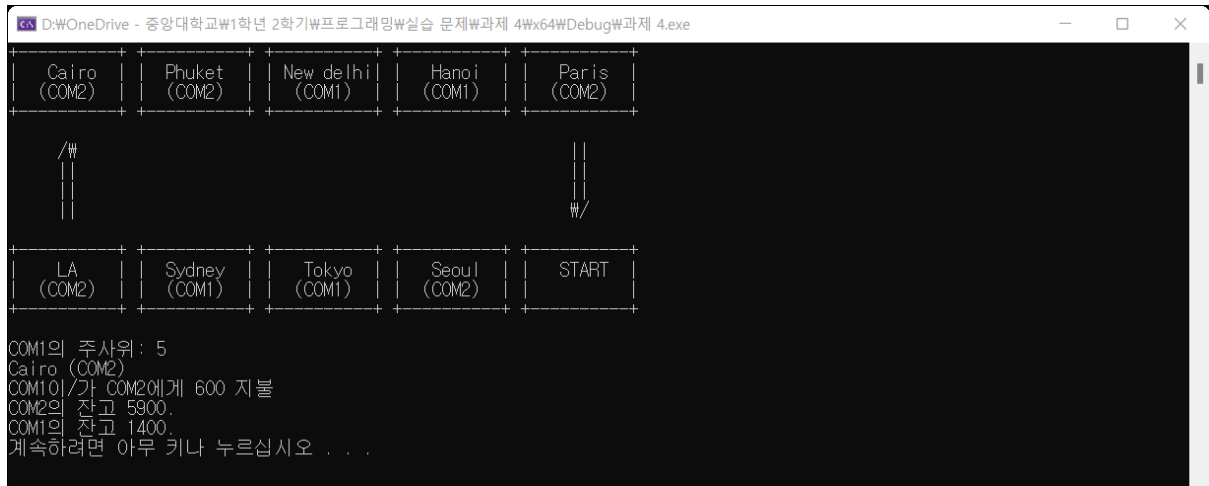


그림 62

그림 62의 화면은 COM1의 마지막 턴이다.



그림 63

그림 63의 화면은 COM2의 마지막 턴이다. 마지막 플레이어의 마지막 턴이 끝났어도 “계속하려면 아무 키나 누르십시오 . . .”라며 계속 게임이 진행되는 것처럼 프로그램이 대기하는데, 이는 모든 턴이 끝났을 때에 대한 별도의 예외처리를 하지 않았기 때문이며, 어차피 이 상태에서 아무 키나 누르면 게임이 그대로 종료되므로 프로그램이 대기하는 것에 대해 특별한 의미는 없다. 물론 턴 숫자가 표시되지 않기 때문에 다음으로 진행할 때까지 플레이어들이 다음 턴의 존재 여부를 모르도록 한다는 점에서 의미는 존재할 수 있으나 그 이상의 의미를 가지지 못한다. 여기서 아무 키나 누를 경우 그림 64의 화면이 나온다. 그림 64의 경우 그림 63에서 화면을 초기화하지 않고 계속 출력한 것으로, 지정된 턴 수가 모두 경과하여 게임을 종료한다는 메시지를 출력하고, 종료 시점에서의 각 플레이어 별 잔고를 출력한 뒤 프로그램을 종료한다.

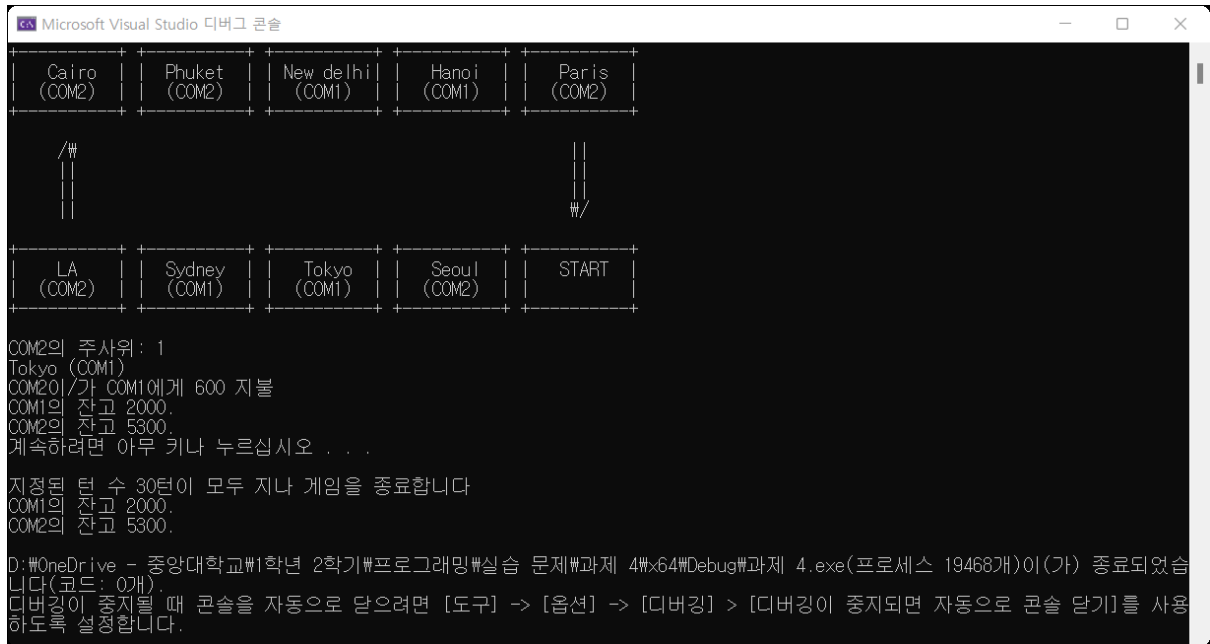


그림 64

이번 게임은 두 플레이어 모두 파산 없이 게임이 종료되었다. 플레이어 중 하나가 파산하여 게임이 종료되는 경우의 실행 결과를 담고자 게임을 한 판 더 진행했다. 이 판에서는 일부러 한쪽이 파산했을 때의 결과를 가져오기 위해 게임 후반부에서 break point를 이용해 일부러 주사위 눈의 값을 직접 조정하였다. 아래는 그 한 판 더 진행한 게임의 결과 화면이다.



그림 65

역슬래시는 이번에도 한국 화폐 단위로 보이는 상황이다.



그림 66



그림 67

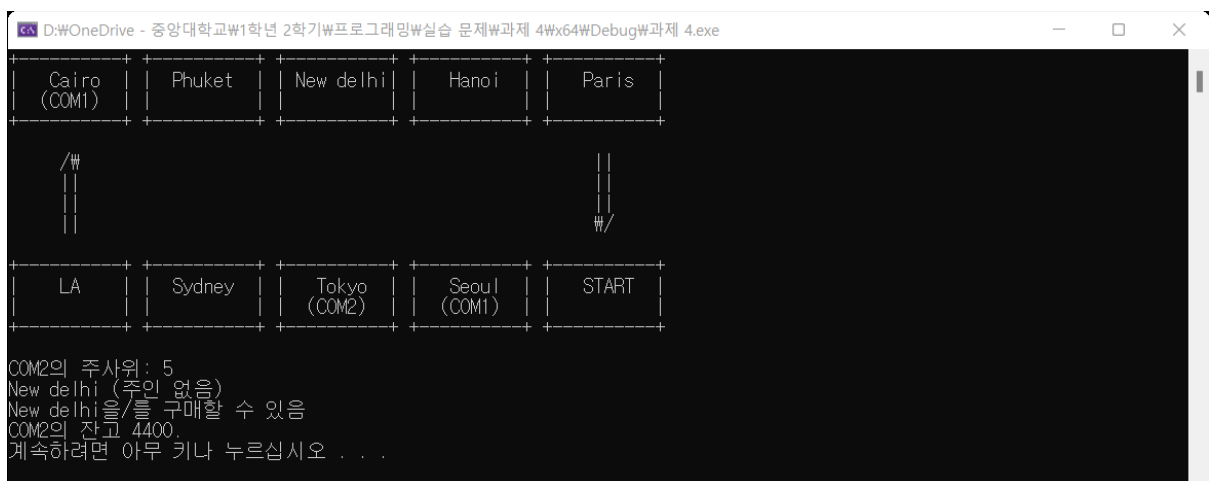


그림 68

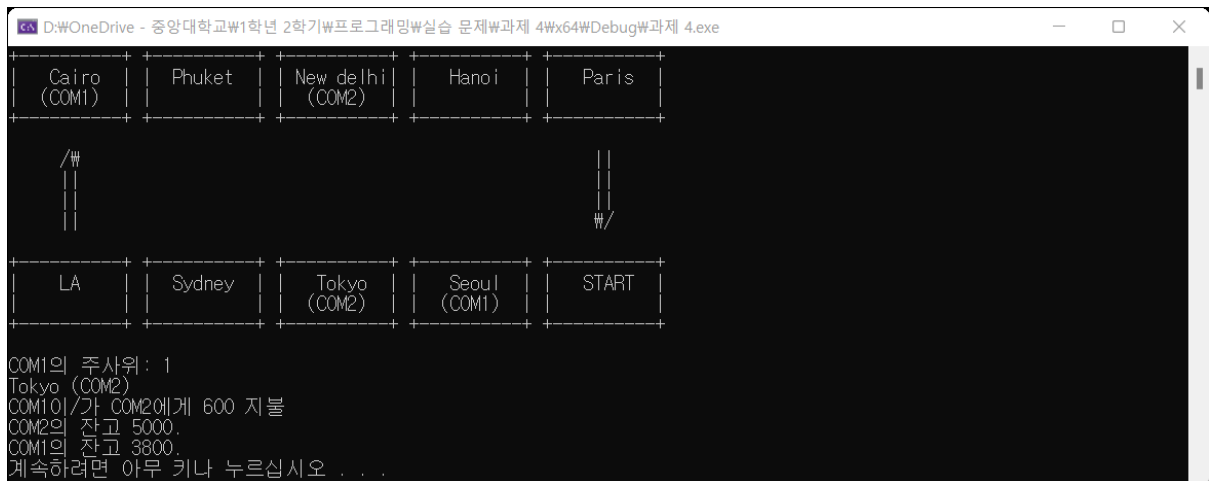


그림 69



그림 70



그림 71

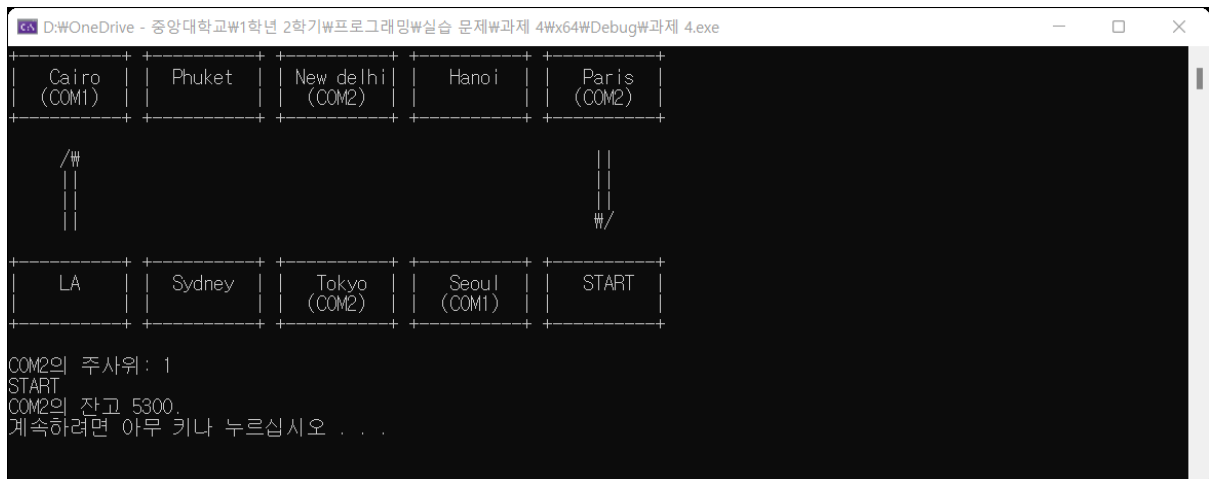


그림 72

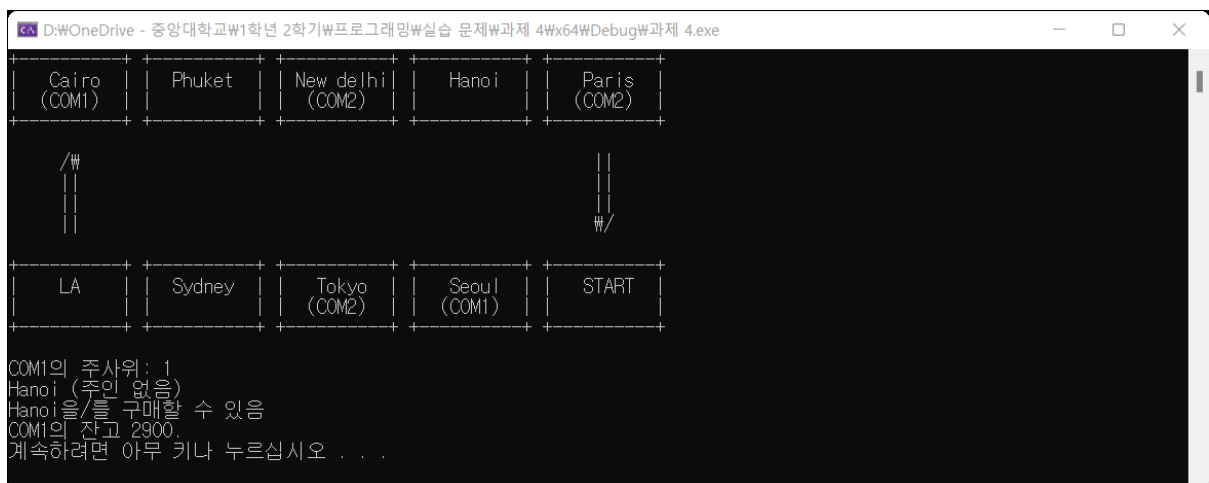


그림 73

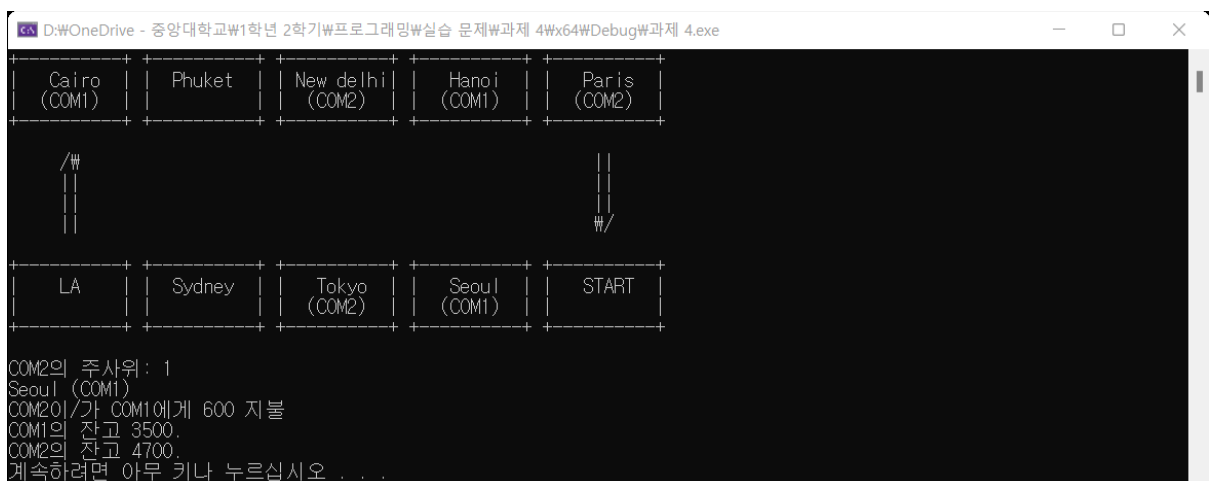


그림 74





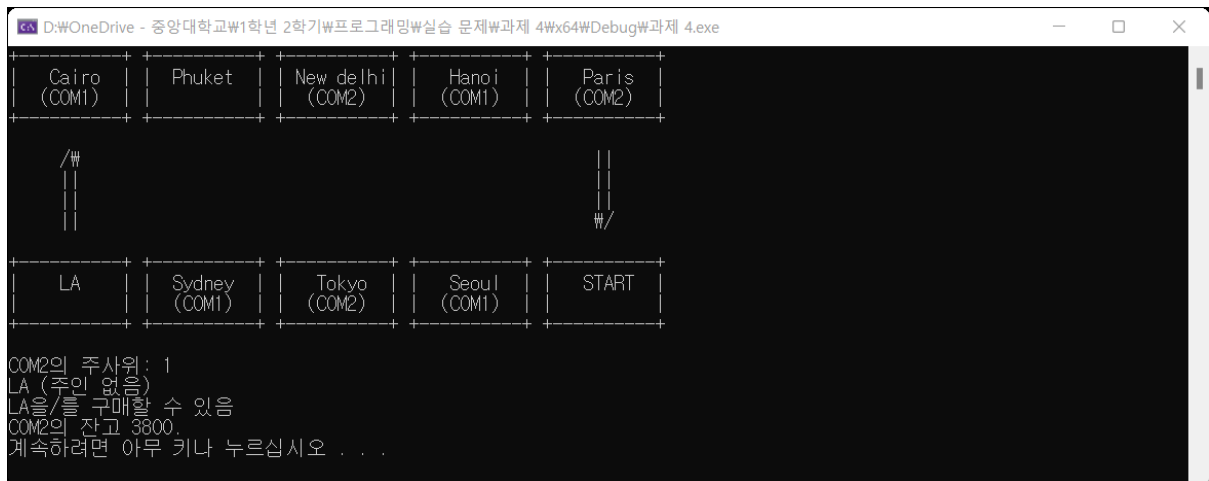


그림 78



그림 79

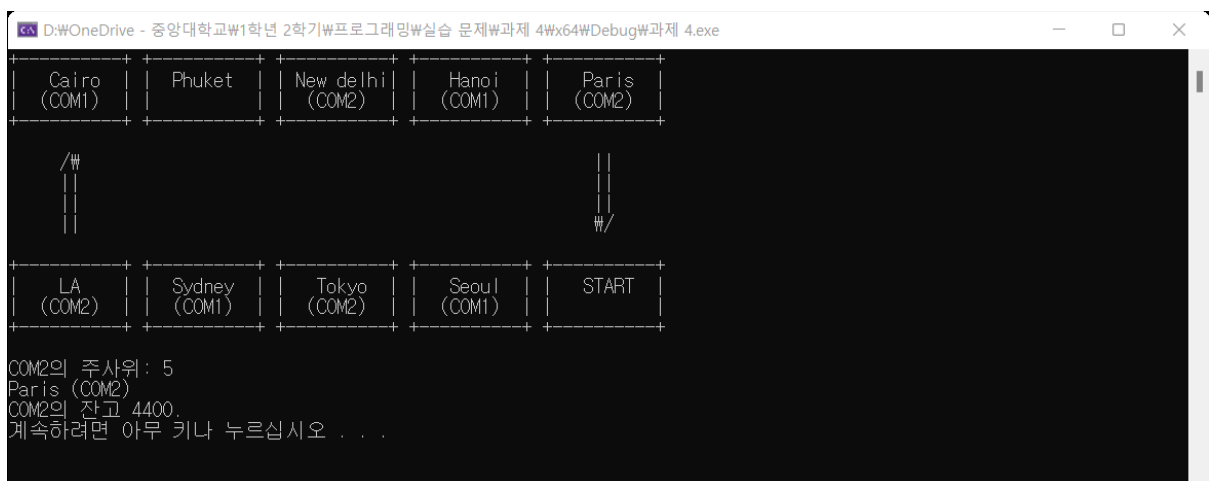


그림 80

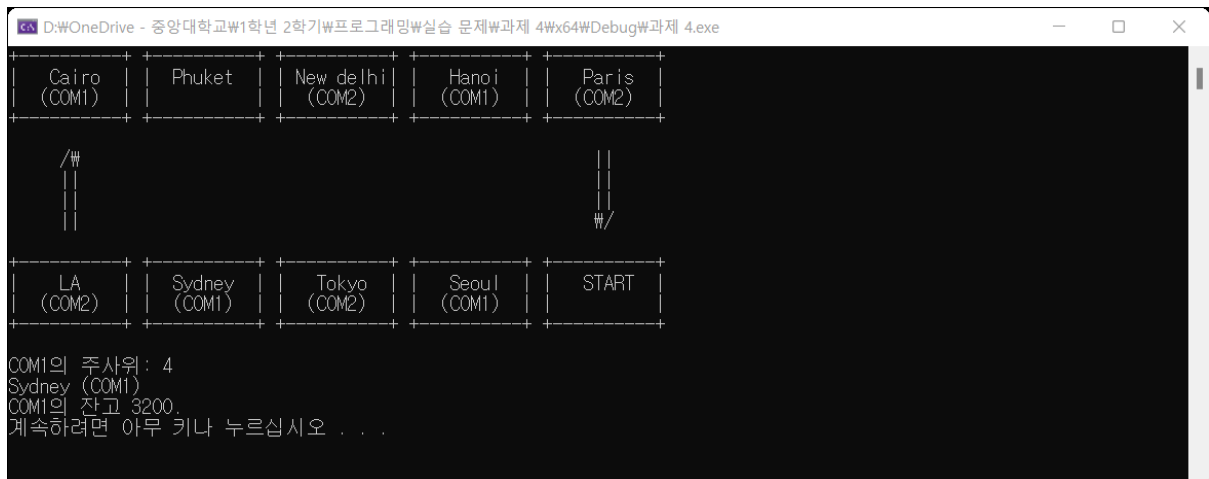


그림 81



그림 82

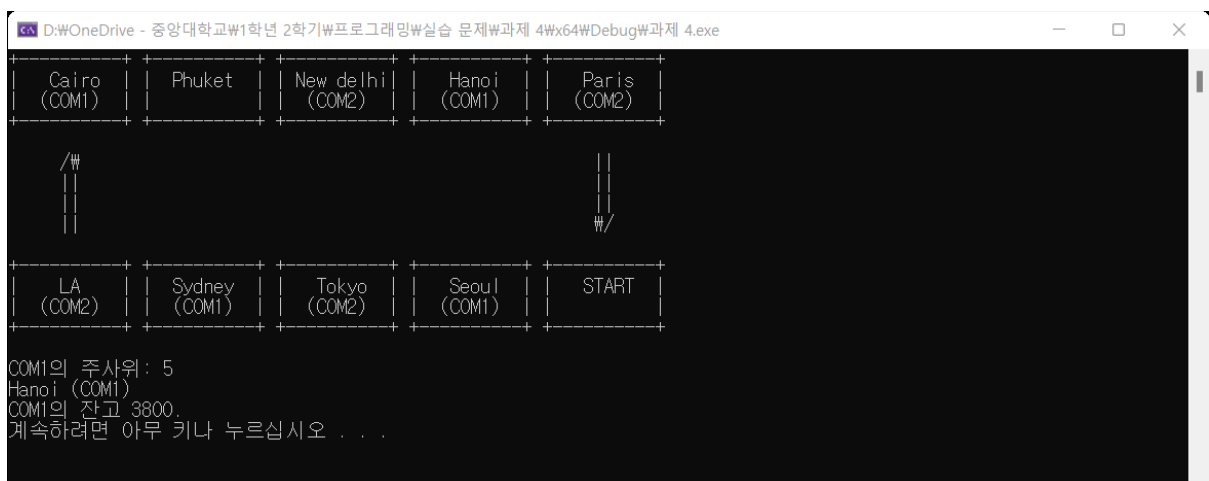


그림 83



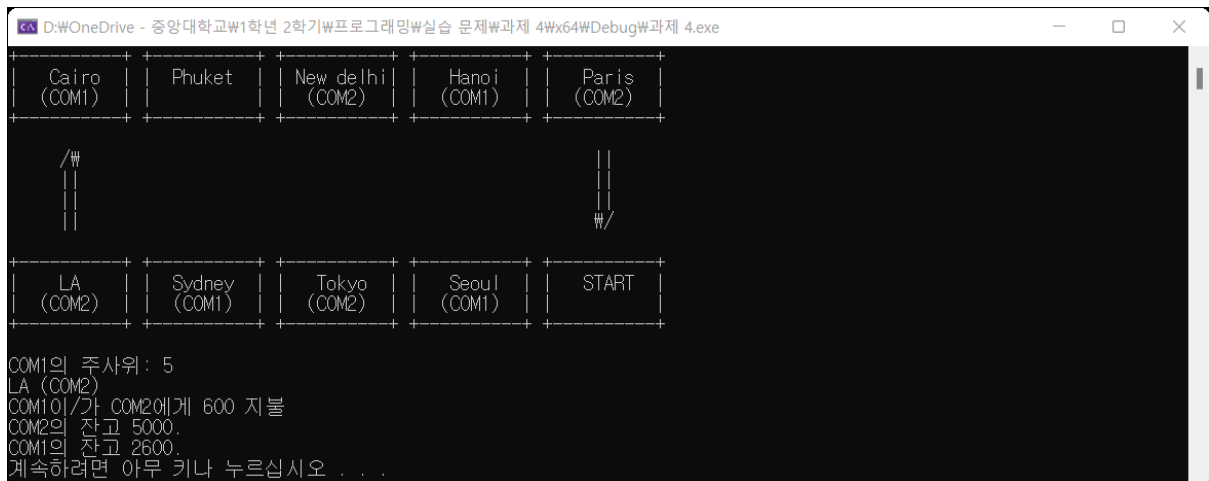


그림 87



그림 88

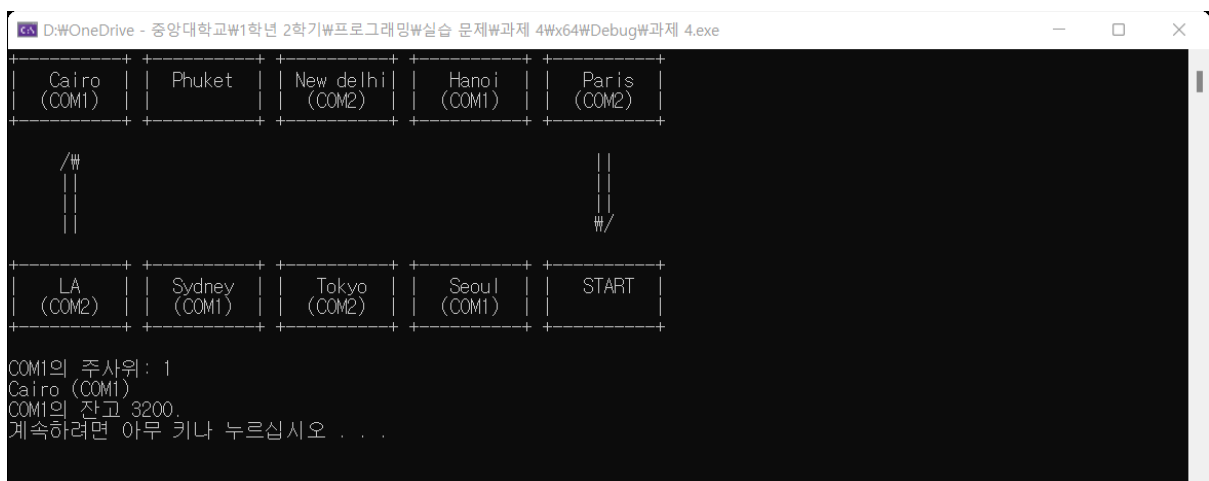


그림 89

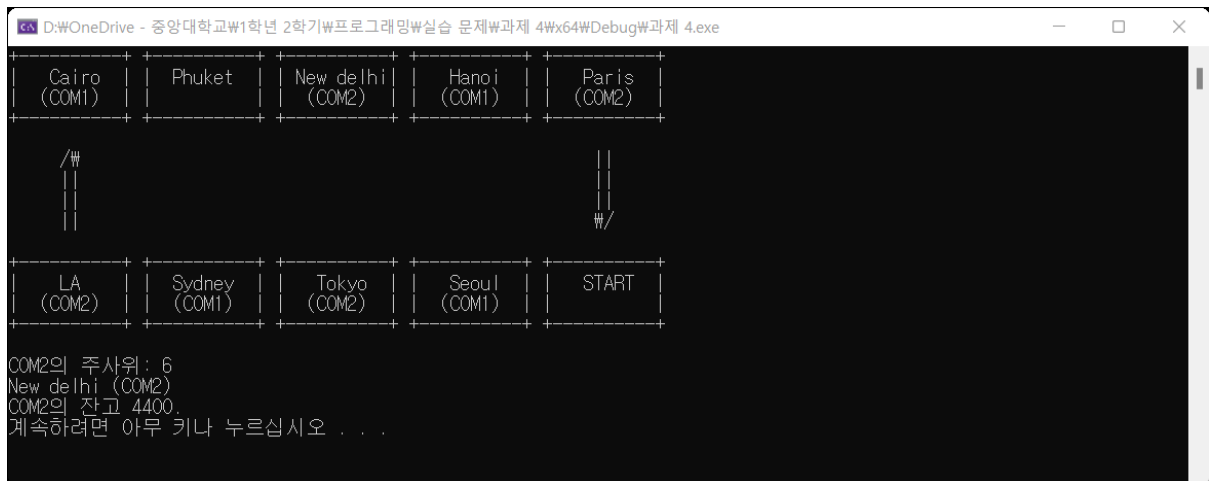


그림 90

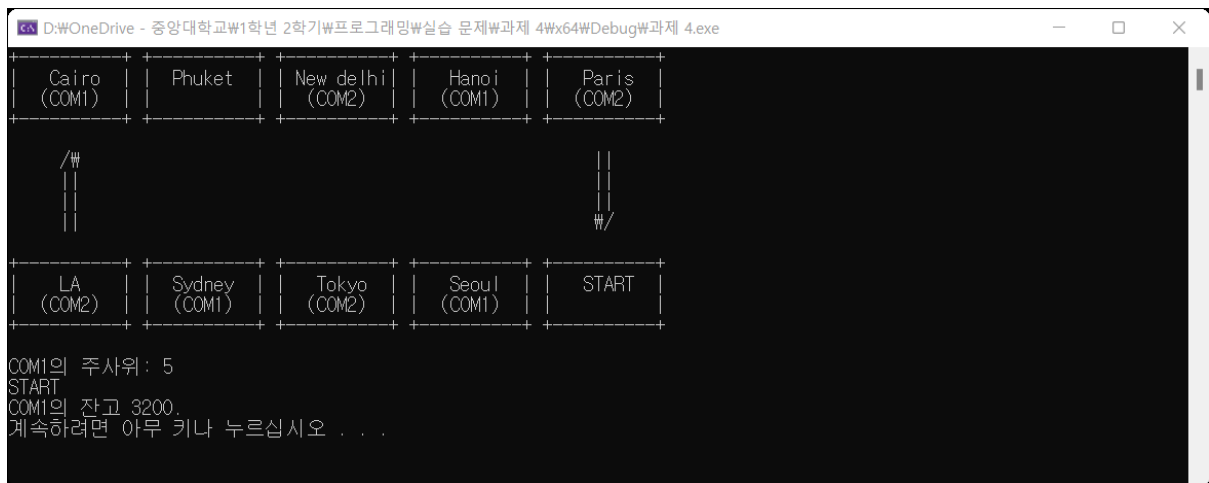


그림 91

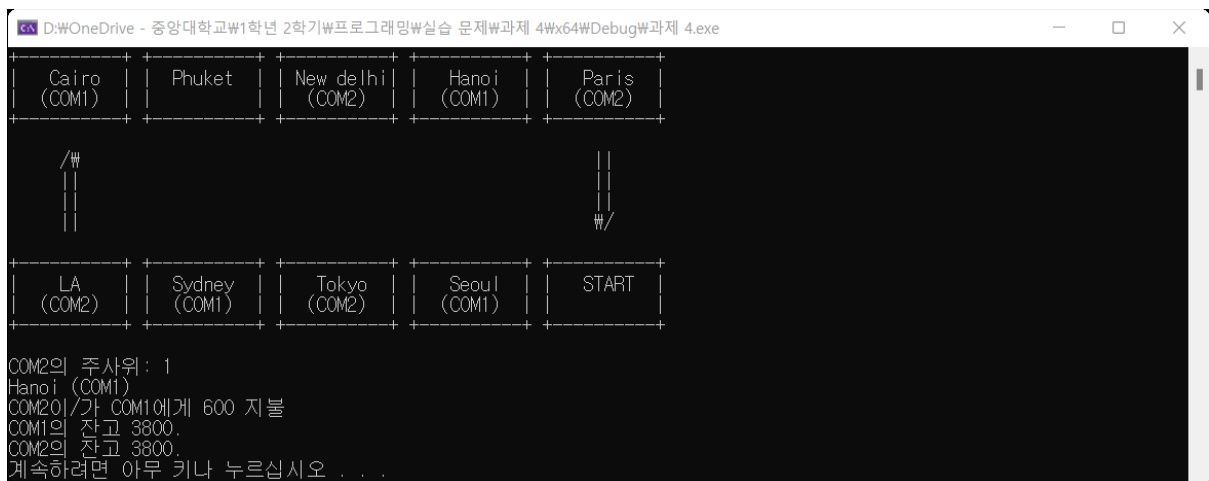


그림 92

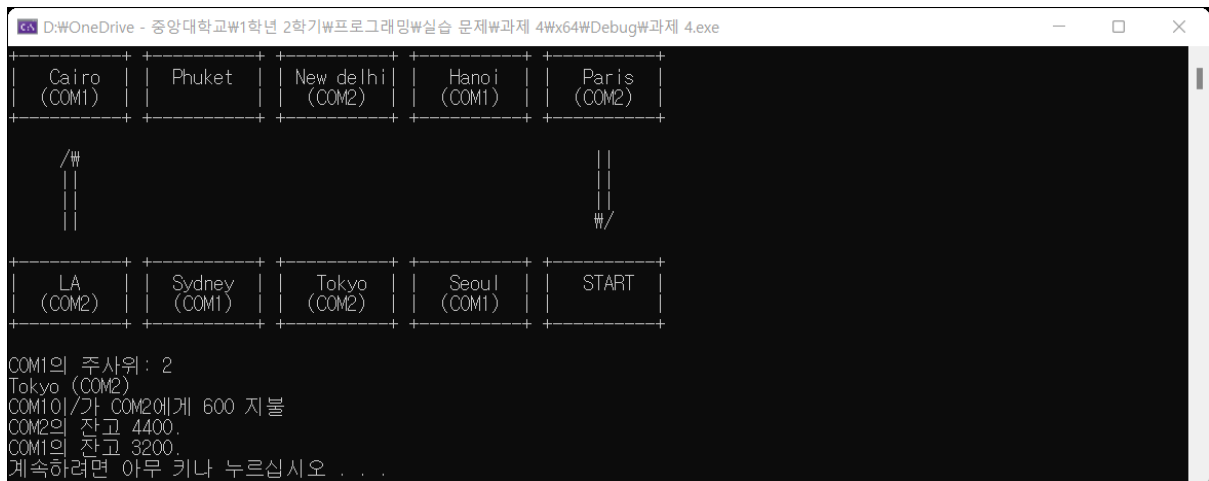


그림 93

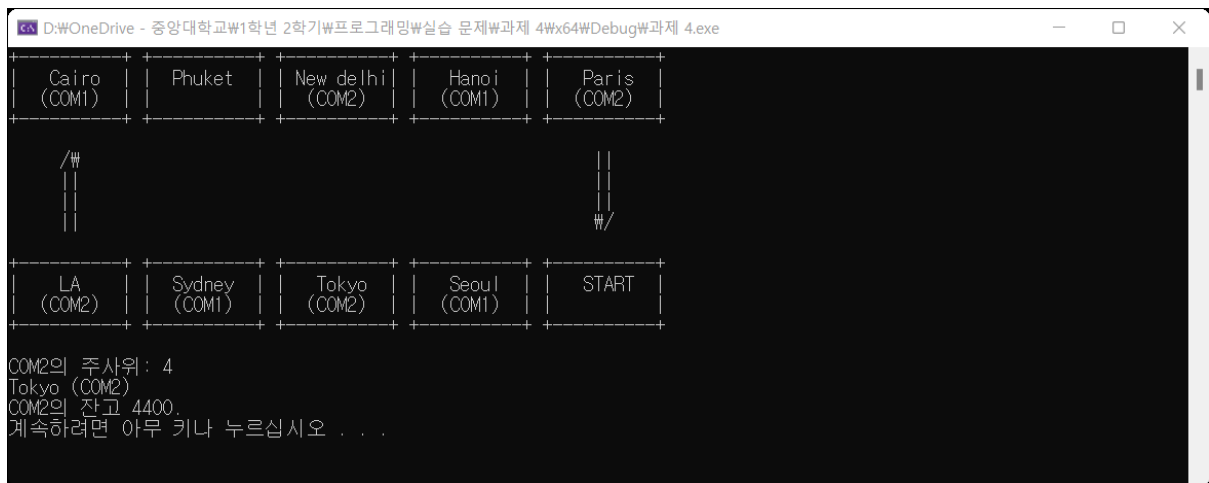


그림 94

그림 94까지 게임이 진행된 결과 설정된 턴의 절반인 15턴이 경과하였다. (그림 30개 / 2 플레이어 = 15턴) 그러나 15턴이 경과하는 동안 Phuket에는 두 플레이어 중 누구도 도착하지 않았다.



그림 95

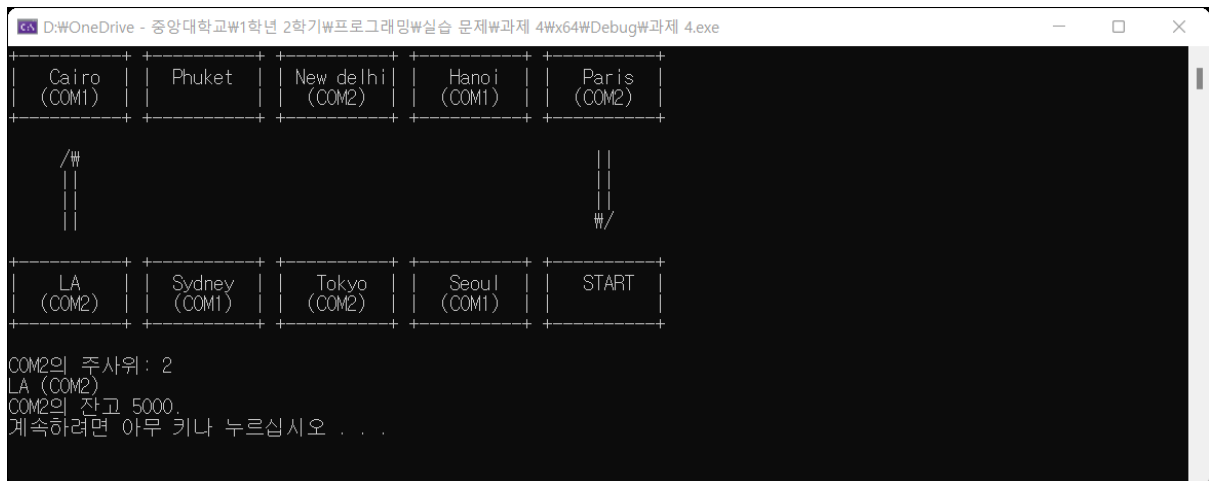


그림 96



그림 97

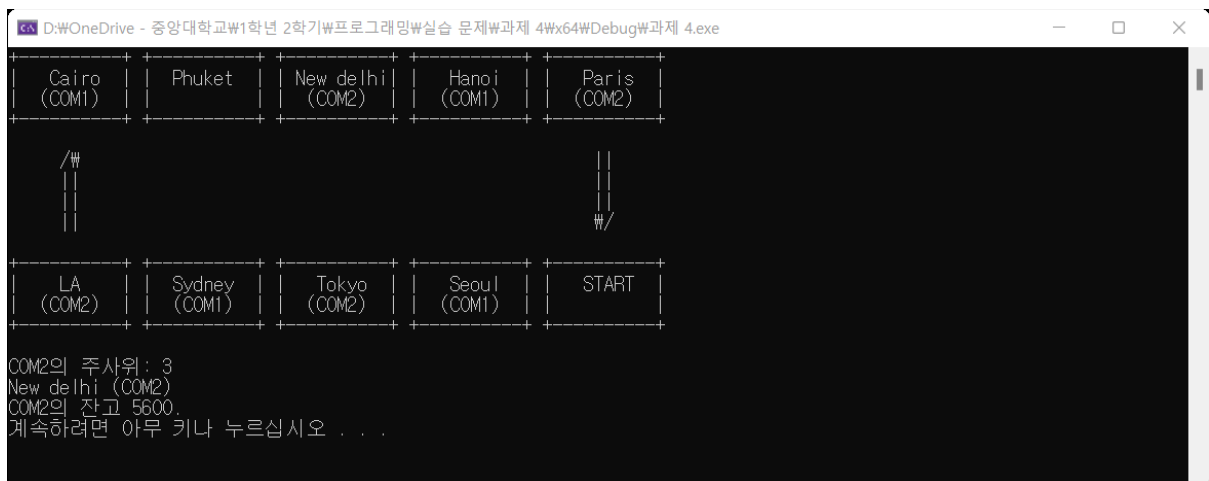


그림 98

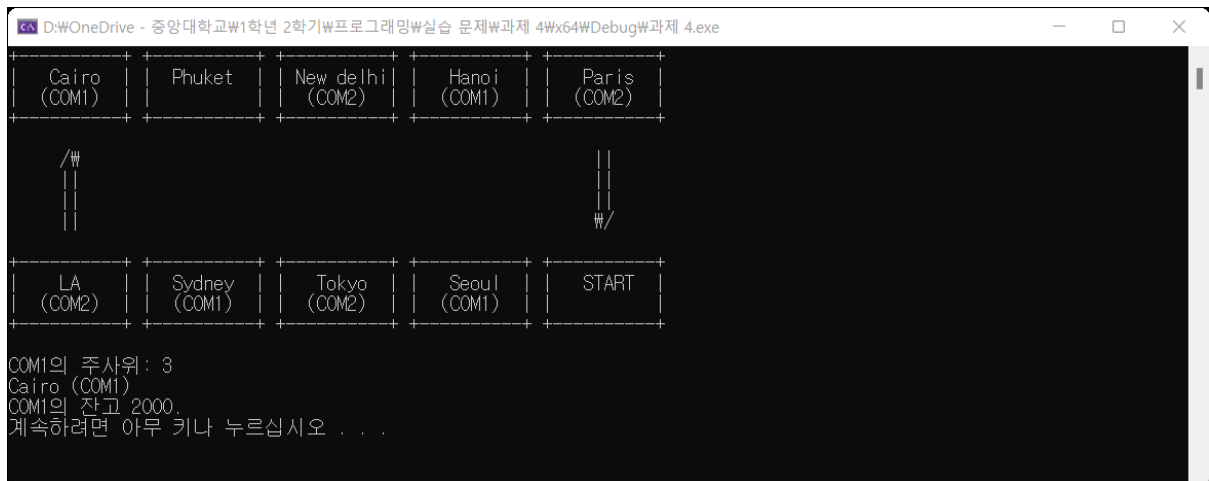


그림 99

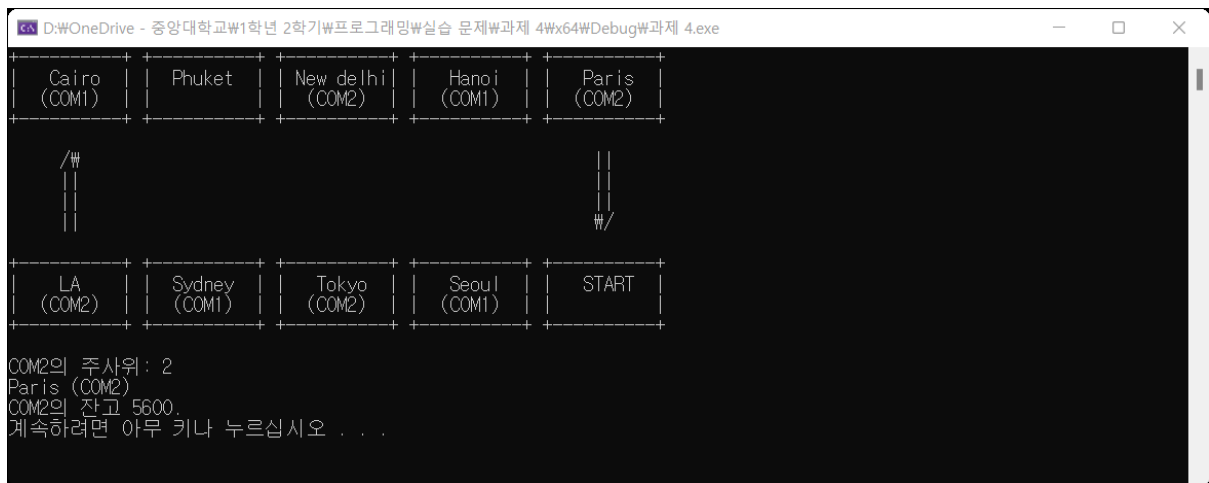


그림 100

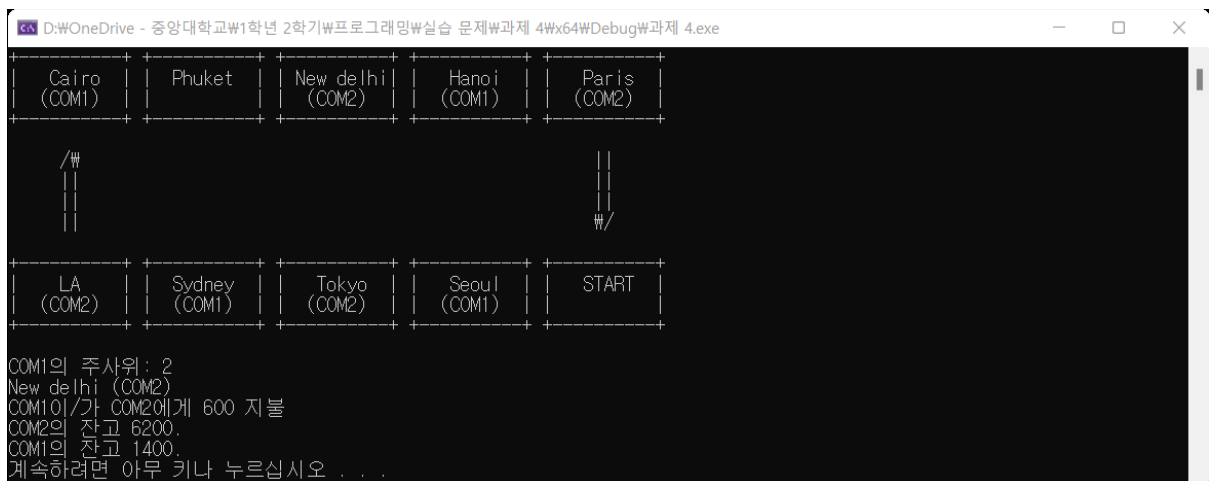


그림 101



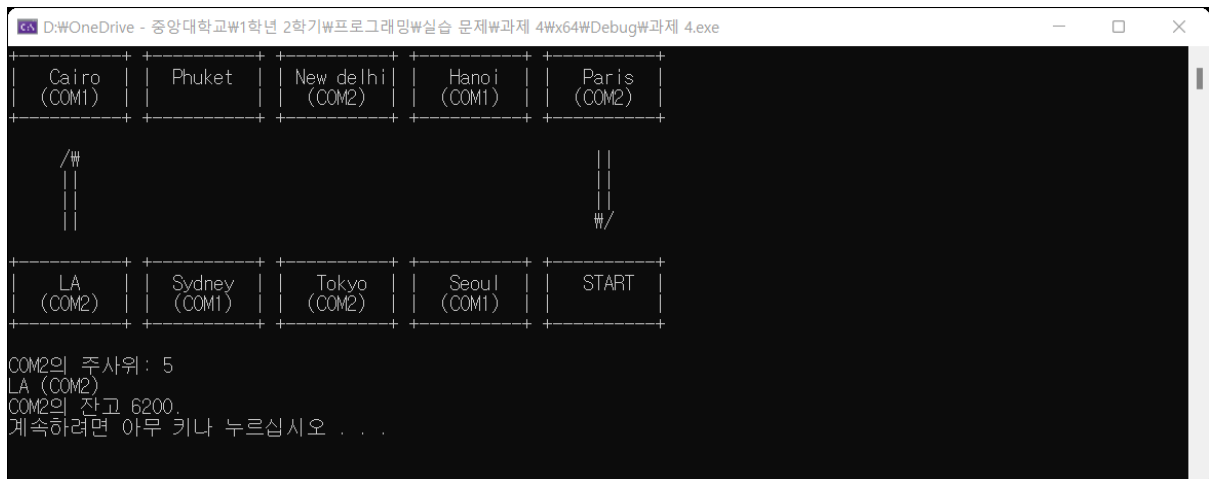


그림 102

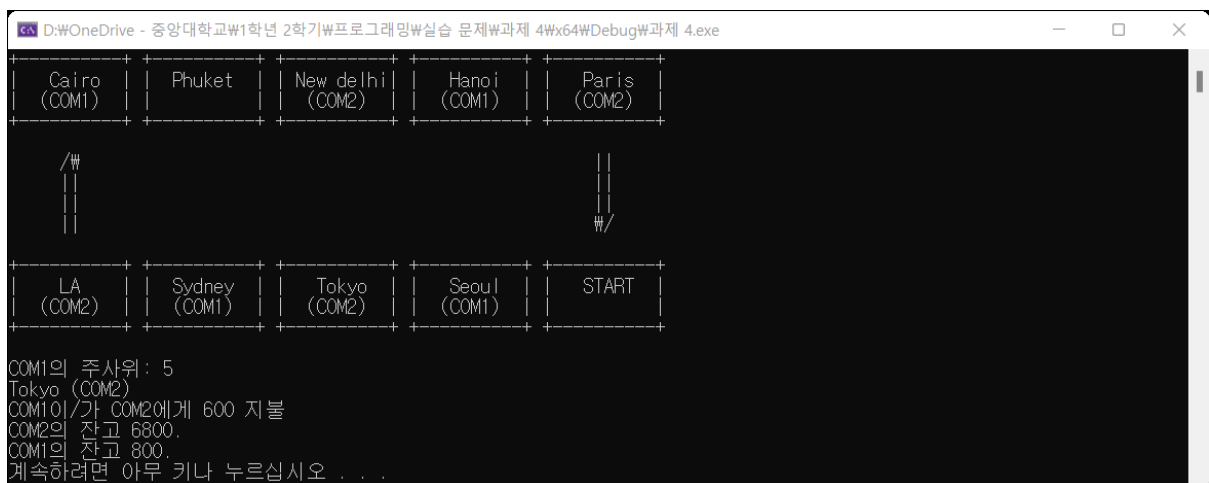


그림 103

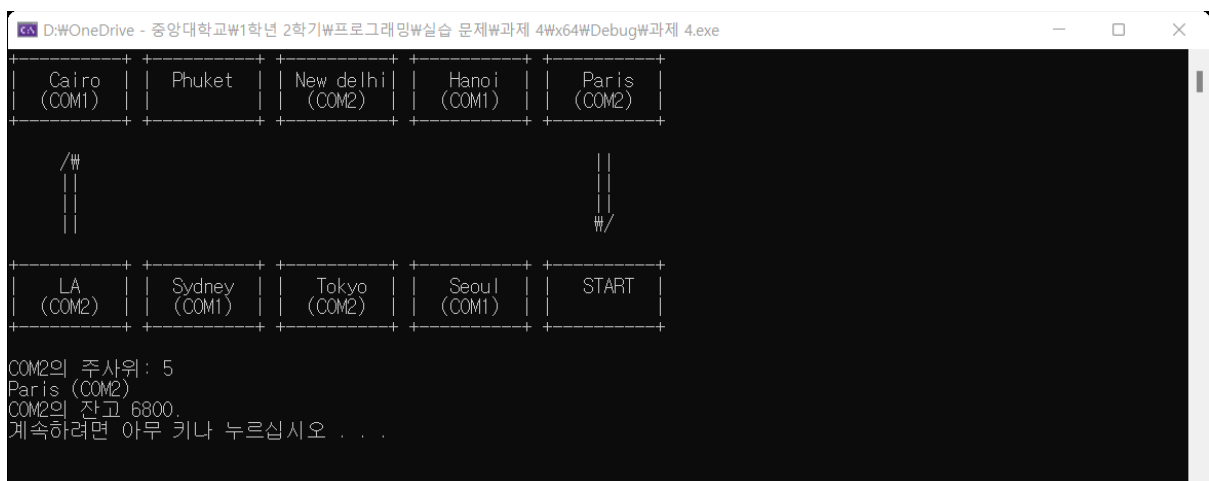


그림 104

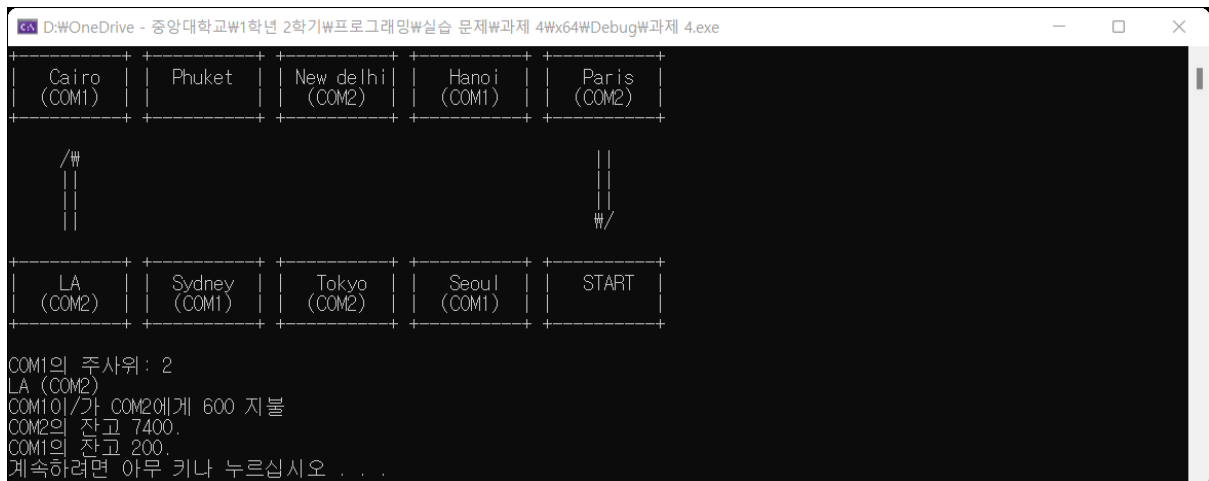


그림 105

그림 105에서 COM1의 현재 잔고는 200원이다. 따라서 이제는 건물도 구매할 수 없으며 통행료를 한 번 더 지불해야 하는 상황이 오면 그 즉시 게임이 종료된다.



그림 106



그림 107

그림 107은 잔고가 부족해 땅을 구매할 수 없는 때의 화면 출력 결과이다. 게임은 계속 진행되나 COM1은 Phuket을 구매하지 못한다.

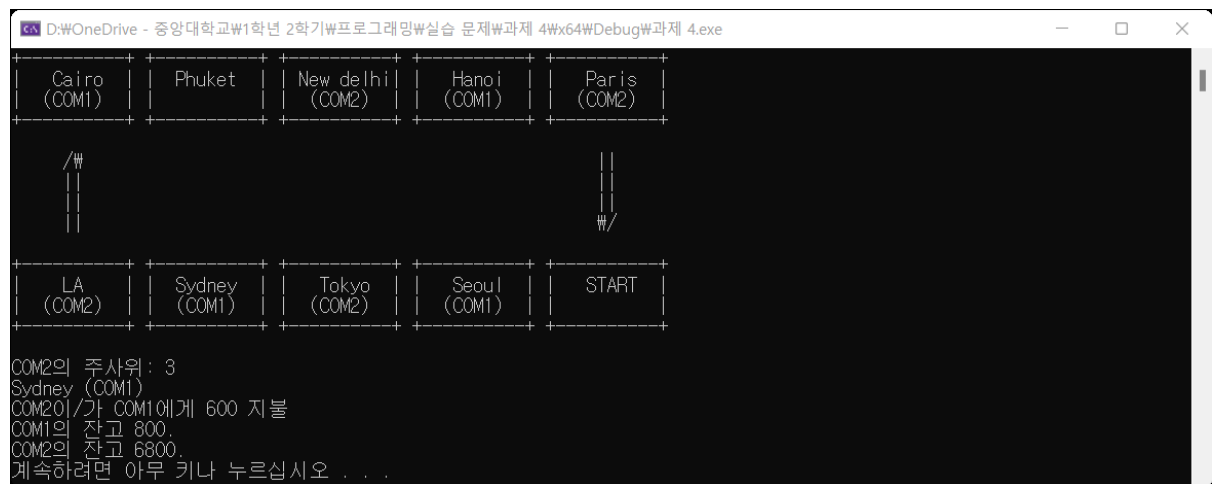


그림 108



그림 109

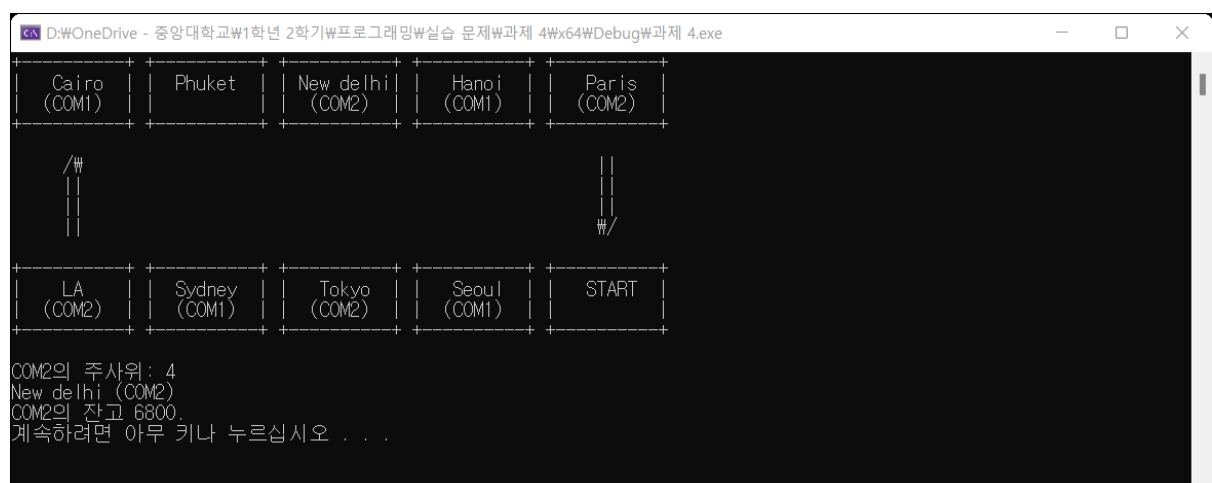


그림 110

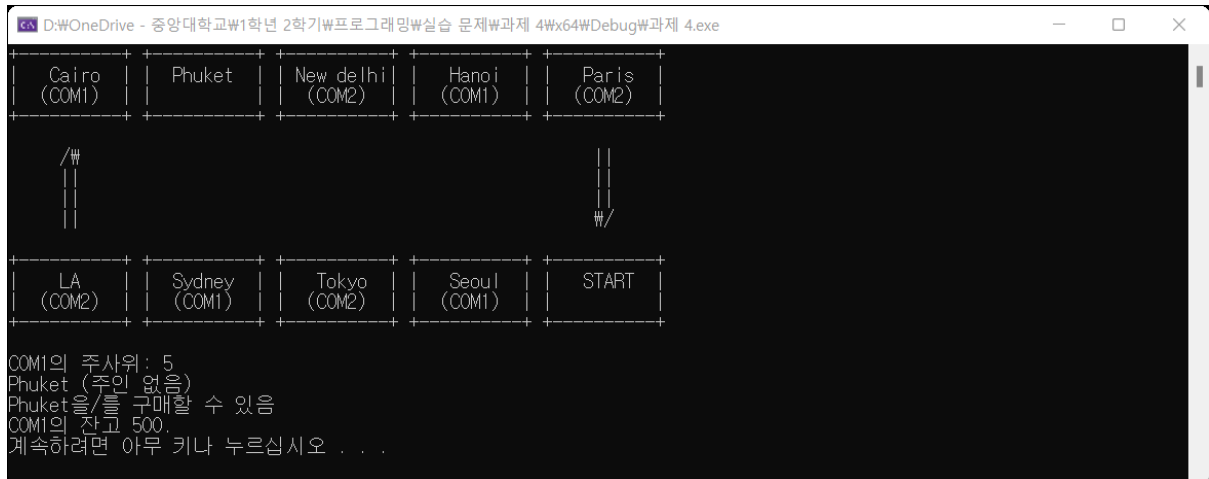


그림 111

그림 111에서 COM1이 이전 턴에 돈이 부족해서 놓쳤던 Phuket 구매에 성공했다. 그러나 이로 인해 잔고가 600 미만으로 낮아지면서 한 번만 통행료를 지불해야 하는 상황에 처해도 파산하는 상태가 되었다.

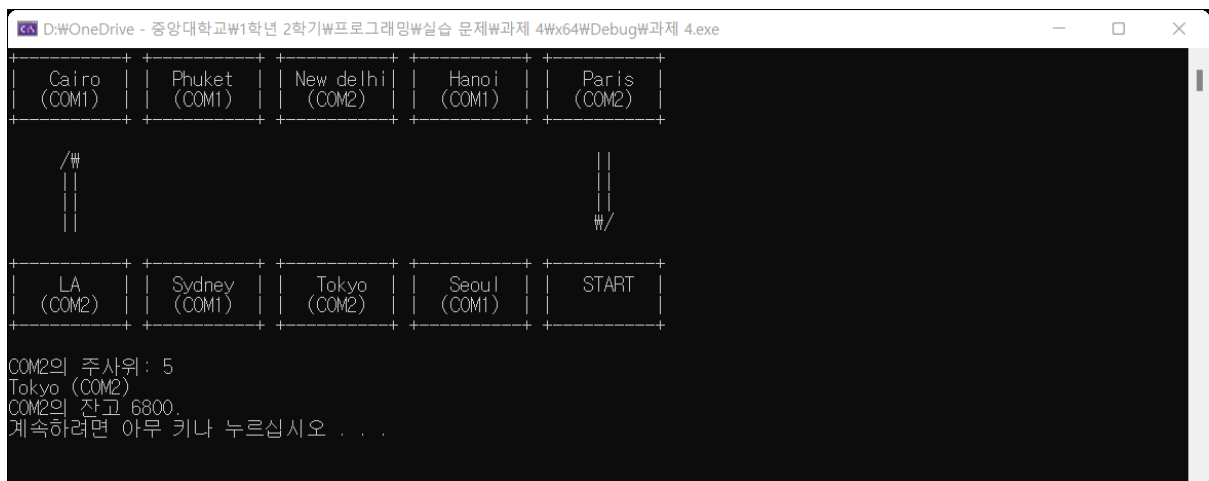


그림 112

그림 111에서 COM1이 구매한 Phuket이 보드에 반영되었다. 이제 모든 땅이 주인을 찾아가게 되었으며 더 이상 땅 구매는 일어나지 않는다.

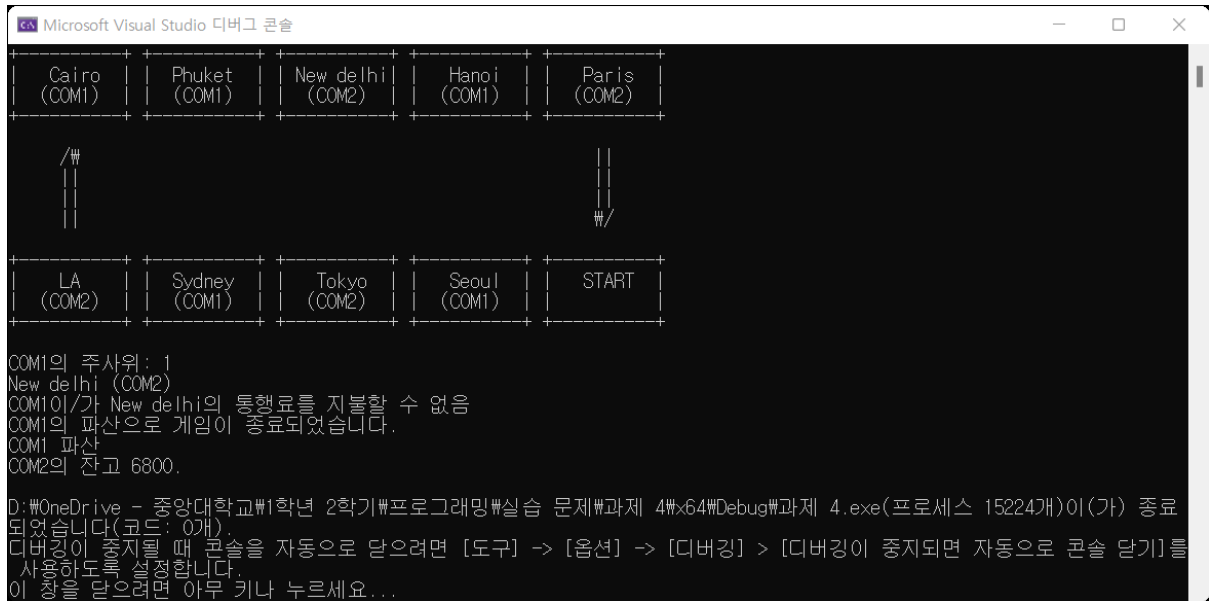


그림 113

결국 COM1이 통행료를 지불해야 하는 상황에 직면하여 게임이 종료되었다. 파산으로 인한 게임 종료시에는 프로그램이 별도로 대기하지 않고 프로그램을 즉시 종료한다. 통행료를 지불할 수 없는 상황에서 통행료 지불이 불가하다는 메시지, 파산으로 게임이 종료되었다는 메시지, COM1의 잔고를 출력하지 않고 파산 정보만 출력, 그리고 파산 즉시 게임이 종료된다는 점 모두를 위 그림 113에서 확인할 수 있다.

돈의 가감이 300의 배수 단위로 이루어지는 특성상 잔고가 정확히 0이 될 수는 없기에, 코드는 잔고가 정확히 0이 되는 상황을 대비하여 완성하였으나 해당 상황에 대해 별도의 테스트는 실시하지 않았다.

## 7. 문제 3

도서 관리 프로그램을 작성한다. 도서 관리 프로그램에는 수많은 책이 등록되어 있고 이 책들은 모두 대출 가능하다. 단 같은 책은 1권만 있다. 책 대여기간은 7일이며, 책 목록과 상태 표시, 대출처리, 반납처리, 대출 목록을 표시하는 프로그램을 작성한다. 그 외의 편의 기능은 넣지 않으며, 회원과 책은 각각 100명 또는 권을 넘지 않는다.

## 8. 문제 3 해결 방안

먼저 각 정보를 저장할 구조체를 선언했다. 대출한 날짜와 반납한 날짜에 활용할 datetime 구조체는 과제 2 문제 1에 사용했던 것을 가져와 약간만 수정해서 사용했고, 책의 정보를 저장할 구조체는 책 이름, 저자명, 출판사, 대출 상황, 그리고 linkedlist 이용을 위한 이전 책 저장 위치,

다음 책 저장 위치를 저장할, 포인터 변수 총 6개를 멤버 변수로 하였고, 회원 정보를 저장할 구조체는 이름과 휴대전화 번호, 그리고 똑같이 linkedlist 이용을 위한 회원 정보 2개의 저장 위치를 저장할, 포인터 변수 총 4개를, 대출 기록의 경우 linkedlist 이용을 위한 대출 기록 2개의 저장 위치를 저장할 변수와 책 및 대출자 정보를 저장할 포인터 변수 2개, 그리고 대출일과 반납일을 저장한 변수로, 총 포인터 변수 4개와 datetime 구조체 2개를 멤버 변수로 하였다. 이때 linkedlist 이용을 위해 다음 구조체 저장 위치뿐만 아니라 이전 구조체 저장 위치까지 저장한 것은, main 함수에서 각 구조체를 가리키는 포인터 변수가 꼭 시작 위치를 가리키지 않을 때를 대비한 것이다. 즉 대출 기록을 제외하고는 지금 당장 큰 의미가 없이 복잡해지지만 했으나 확장성을 고려한 것이다. (대출 기록은 마지막에서부터 찾는 기능이 있어 이를 활용한다.) 우선 테스트로 5개의 책과 회원만을 등록하므로 이를 사용자에게서 입력받는 게 아니라 직접 초기화해 주었다. 각 5개씩을 선언할 때 배열로 선언하였으나, 배열 자체가 linkedlist를 고려해 만들어졌기에 이전 구조체 저장 위치와 다음 구조체 저장 위치까지 직접 값을 넣어주었다. 대출 기록의 경우 없는 것으로 시작하므로 별도의 배열을 선언하지 않고 포인터로 선언했으며, NULL로 초기화해 주었다.

이후 화면에 기능 목록을 표시하고, 기능에 대한 번호를 입력받아 각 기능을 수행한다. 1이 입력된 경우, 책 구조체 포인터를 전달받아 책 목록을 표시한다. 표시하기 전 책이 있는지 확인하고 linkedlist로 연결된 책 구조체 포인터 중 가장 앞에 있는 것을 찾는다. 이는 앞서 구조체를 정의할 때 전달받은 포인터 변수가 시작 위치가 아닌 것을 고려하였으므로, 이에 맞추는 것이다. 가장 앞에 있는 것을 찾았으면, 순서대로 제목과 저자, 출판사와 상태를 표시한다. 제목, 저자, 출판사는 왼쪽 정렬했으며, 글자 수 상한은 각각 20byte로 하였다. 대출 중인 것과 보유 중인 것을 구분하는 방법은, 대출 상황을 저장하는 포인터 변수가 NULL을 가리키고 있으면 대출 중인 것, 그렇지 않으면 보유 중인 것으로 구분해 출력했다. 이 출력 또한 앞 문제처럼 문자열에 모아두었다가 한꺼번에 출력했다.

2가 입력된 경우 개행 문자를 받아 버려준 후 대출을 시행한다. 먼저 회원이나 대출할 수 있는 책이 있는지 확인하고 대출 기록을 저장할 공간을 동적으로 할당 받는다. (회원별 대출 제한과 관련해서는 제한이 없으므로 대출할 수 있는 회원이 있는지는 확인하지 않는다.) 이후 대출할 회원의 이름을 입력받아 해당 회원을 찾는다. 이와 비슷하게 대상이 되는 책도 찾는다. 이때 같은 책은 1권만 있으므로 이름이 같아도 해당 책의 대출 정보가 있다면 잘못된 입력으로 보았다. 이후 대출 날짜를 입력받은 후 책 구조체에 대출 기록을 등록하고 다른 대출 정보와 linkedlist를 연결하면 대출 작업은 끝난다. 이때 없는 회원의 이름이나 존재할 수 없는 날짜 등 잘못된 정보를 입력하면 다시 입력해 달라는 메시지와 함께 올바른 정보가 들어올 때까지 다시 입력받았다. 또 회원이나 책을 찾을 때는 linkedlist의 맨 앞부터 하나하나 strcmp를 쓰면서 비교해 나갔다. 또 fgets 함수로 회원이나 책 이름을 입력받을 때 한 번에 최대 회원/책 이름 길이만큼 입력받았는데, 입력받은 문자열이 개행 문자로 끝나지 않으면 아직 입력받을 게 뒤에 남아있고 어느 회원/책에도 일치하지 않는 것으로 봐서, fgets 함수를 계속 호출하며 버퍼를 비우고 다시 입력해 달라는 메시지를 띄웠다. 정상적으로 입력되었으면 strcmp를 쓰기 전 마지막 개행문자를 지워줬다. 이를 지워주

지 않으면 개행문자의 차이로 인해 같은 문자열도 다른 것으로 판정되기 때문이다.

3이 입력된 경우도 마찬가지로 개행 문자를 받아 버려준 후 반납을 시행한다. 똑같이 먼저 반납할 수 있는 책이 있는지 확인하고 반납할 책이 있다면 책의 이름을 입력받아 책과 반납 처리할 대출 기록을 찾는다. 이후 반납 날짜를 입력받는데, 이때 반납 날짜가 대출 날짜보다 빠르면 잘못된 입력으로 보고, 년도부터 다시 입력하도록 했다. 날짜까지 정상적으로 입력되었으면 반납날짜를 대출 기록에 저장하고 책 구조체에서 대출 기록을 가리키는 포인터 변수를 NULL로 초기화(즉 대출 중이지 않다는 정보를 표현한 것)함으로써 반납 작업을 끝낼 수 있다. 이때 대출 기록은 다른 기록과 달리 뒤에서부터 찾는데, 이는 반납되지 않은 대출 기록이 나중에 있을 확률이 높다는 점을 이용해 찾는 시간을 단축하려 하는 것이다.

4가 입력된 경우 1과 마찬가지로 대출 기록 구조체 포인터를 받아 하나씩 화면에 표시한다. 맨 앞에 있는 대출 기록을 찾고, 각 기록의 책 이름, 대출자 이름과 전화번호, 대출 날짜와 반납 날짜를 표시한다. 이때 대출중인 경우 반납날짜 대신 해당 사항을 표기하고, 날짜 표시의 경우 `yyyymmdd` 형식으로 표기한다. 여기서 `yyyymmdd`는 예를 들면 2023년 9월 1일의 경우 20230901로 표시하는 형태이며, 년도가 4자리인 경우 총 8자리라서 `int` 형으로 충분히 표기가 가능하나 10000년을 넘는 상황을 대비해 년도를 9자리 정도(정확히는 `unsigned int` 형으로 표기할 수 있는 최댓값)까지 표기할 수 있도록 `unsigned long long` 형으로 표기했다. 이때 대출 기록이 없는 경우 대출 기록이 없다는 메시지를 띄우고 끝냈다.

4가지 입력 이외의 경우, 개행 문자 등 일부 문자는 별도의 메시지 출력이 없도록 했고, 이외에는 올바른 입력이 아니니 다시 입력해달라는 메시지를 출력하고 기능 목록을 다시 표시한 뒤 다른 숫자 입력을 기다렸다.

이를 이용해 문제 3을 해결한 결과는 다음과 같다.

## 9. 문제 3의 결과

“한계테스트용”으로 시작하는 문자열은 모두 사용할 수 있는 문자열의 길이를 최대로 채운 것으로, 해당 경우에 문제가 없는지 확인하는 용도이다.

원래 화면을 초기화하지 않는 프로그래머라 한 번의 캡처만이 필요하나, 내용이 너무 길어 여러 개의 캡처본으로 나누었다. 아래 그림 115~119는 프로그램의 실행 결과를 나타낸 것이다.

아래 캡처본에서 회원의 기본값(초기화 값)은 다음과 같다. 책의 정보는 그림 114 바로 위에서 확인할 수 있는데 회원은 그렇지 못해 별도로 작성한다.

이름	전화번호
이순신	010-4444-5555
도라에몽	010-3333-8888

나상실	010-3434-3434
김승태 교수님	010-4757-0040
한계테스트용가상이름	한계테스트용가상번호

표 114



그림 115





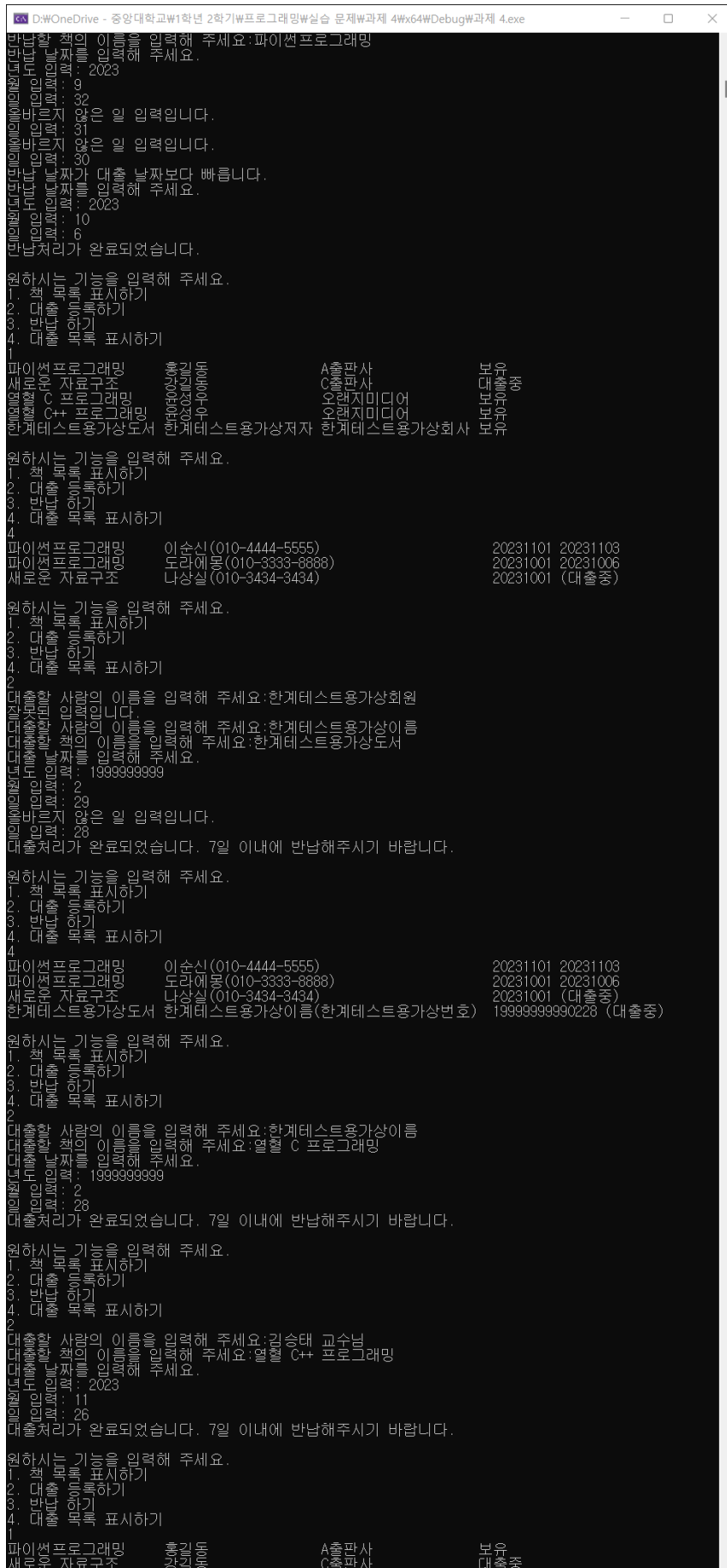


그림 117

```
D:\OneDrive - 중앙대학교\1학년 2학기\프로그래밍\실습 문제\문제 4\Wx64\Debug\문제 4.exe
열월 C 프로그래밍 윤성우 오렌지미디어 대출중
열월 C++ 프로그래밍 윤성우 오렌지미디어 대출중
한계테스트용가상도서 한계테스트용가상저자 한계테스트용가상회사 대출중

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
4
파이썬 프로그래밍 이순신(010-4444-5555) 20231101 20231103
파이썬 프로그래밍 도라에몽(010-3333-8888) 20231001 20231006
새로운 자료구조 나상실(010-3434-3434) 20231001 (대출중)
한계테스트용가상도서 한계테스트용가상이름(한계테스트용가상번호) 199999999990228 (대출중)
열월 C 프로그래밍 한계테스트용가상이름(한계테스트용가상번호) 199999999990228 (대출중)
열월 C++ 프로그래밍 김승태 교수님(010-4757-0040) 20231126 (대출중)

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
2
대출할 사람의 이름을 입력해 주세요:나상실
대출할 책의 이름을 입력해 주세요:파이썬프로그래밍
대출할 날짜를 입력해 주세요.
년:2024
월:1
일:1
반납처리가 완료되었습니다. 7일 이내에 반납해주시기 바랍니다.

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
1
파이썬 프로그래밍 홍길동 A출판사 대출중
새로운 자료구조 김민준 C출판사 대출중
열월 C 프로그래밍 윤성우 오렌지미디어 대출중
열월 C++ 프로그래밍 윤성우 오렌지미디어 대출중
한계테스트용가상도서 한계테스트용가상저자 한계테스트용가상회사 대출중

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
4
파이썬 프로그래밍 이순신(010-4444-5555) 20231101 20231103
파이썬 프로그래밍 도라에몽(010-3333-8888) 20231001 20231006
새로운 자료구조 나상실(010-3434-3434) 20231001 (대출중)
한계테스트용가상도서 한계테스트용가상이름(한계테스트용가상번호) 199999999990228 (대출중)
열월 C 프로그래밍 한계테스트용가상이름(한계테스트용가상번호) 199999999990228 (대출중)
열월 C++ 프로그래밍 김승태 교수님(010-4757-0040) 20231126 (대출중)
파이썬 프로그래밍 나상실(010-3434-3434) 20240101 (대출중)

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
2
대출 가능한 책이 없습니다.

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
3
반납할 책의 이름을 입력해 주세요:나상실
반납할 책의 이름을 입력합니다.
반납할 책의 이름을 입력해 주세요:새로운 자료구조
반납할 날짜를 입력해 주세요.
년:2024
월:1
일:1
반납처리가 완료되었습니다.

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
3
반납할 책의 이름을 입력해 주세요:열월 C 프로그래밍
반납할 날짜를 입력해 주세요.
년:21000000003
월:6
일:31
출발하지 않은 일 입력입니다.
일:30
반납처리가 완료되었습니다.

원하시는 기능을 입력해 주세요.
1. 책 목록 표시하기
2. 대출 등록하기
3. 반납하기
4. 대출 목록 표시하기
3
반납할 책의 이름을 입력해 주세요:한계테스트용가상도서
반납 날짜를 입력해 주세요.
```



## 10. 소스코드

첨부파일 참조