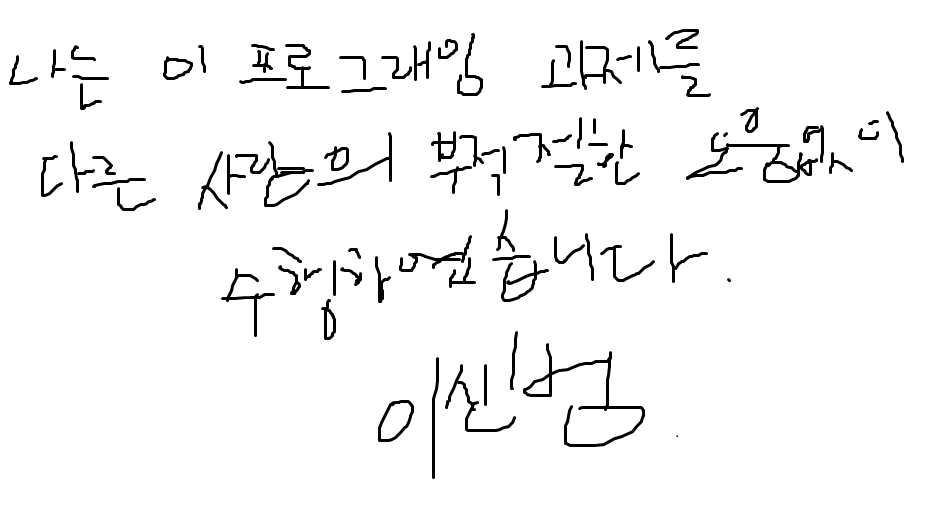
**ASSIGNMENT #5**

지도교수 : 윤은영 교수님

학생 이름 / 학번 / 학과 : 이신범 / 20180285 / 무은재 학부

POVIS ID : nm160

Problem #2

1. 개요

어싸인 5는 링크드 리스트를 이용한 정보의 삽입, 삭제를 응용하여 단방향 환형 연결리스트를 제작하고, 수건돌리기 게임을 시뮬레이션하는 과제였다.

1. 알고리즘

<pseudocode>

Typedef struct student

{

이름

성별

학과

학번

조

}STUDENT;

Typedef struct node

{

STUDENT data;

Int mynum; // 해당 조에서 head -> 1번 그 이후로부터 2번, 3번 ,,,

Node\* next;

}NODE

Typedef struct team

{

Int count;

Node\* head;

}TEAM

// 구조체 TEAM 을 초기화하는 함수 //

Void putjo(TEAM\* 1 2 3 4 5) int count = 0; head=NULL

// players.txt 에서 이름, 성별, 학과, 학번, 조를 콜바이 어드레스로 읽어오는 함수 //

int getstu(FILE\* input,const char\* name,const char\* gender,const char\* dept,const int\* stuid,const int\* jo);

// 읽어온 STUDENT 신상으로 채운 STUDENT를 반환하는 함수 //

STUDENT putstu(char name, char gender, char dept, int stuid, int jo);

// 입력받은 jo와 STUDENT를 통해 NODE (linked list)을 생성하는 함수 // 맨 처음 넣은게 head

// Enqueue를 실행할 때 마다, TEAM의 count ++해주고, NODE->mynum == count 해준다. //

Void linkjo(TEAM\*1 2 3 4 5 , STUDENT data) // 이후 add도 이거 써먹음

{

TEAM->count ++

NODE\* stu;

NODE\* temp = (NODE\*)malloc(sizeof(NODE));

Temp -> data = data

Temp -> mynum = TEAM -> count

If(TEAM->count == 1) // 첫 NODE 생성

TEAM -> head = temp;

Temp -> neat = TEAM -> head

// 이후 NODE 생성하는 경우 마지막 NODE를 찾아 stu에 넣고, 그 뒤에 temp 를 연결시킨다. //

If else(TEAM->count > 1 )

dd

Temp -> next = TEAM -> head

// 입력받은 조의 player 목록을 출력하는 함수 //

Void printjo(TEAM\* 1 2 3 4 5) // for 문을 I = 0; i<count; i++ 으로 돌리면 해당 조 출력가능하다.

// 해당 학번에 해당하는 학생을 찾기 위해 linked list를 탐색하는 함수 // remove에 이용

// 해당 학번에 해당하는 전 node를 반환해야 의미가 있다!! //

// 주솟값 > 0 이므로, 있으면 true, 없으면 false 반환하게 된다. //

NODE\* findnode(TEAM\*\*, int stuid)

// 지우고싶은 NODE의 이전 NODE를 입력받아 지우고 싶은 NODE를 삭제하는 함수 //

// TEAM의 count 변화 등도 생각하는 것은 젤 아래의 REMOVE 함수에서 하자. //

Void removenode(TEAM\* 1 2 3 4 5, NODE\* pPre)

NODE\* Temp = 지우고 싶은 NODE\* 하고 freenode(temp);

// NODE 하나를 free하는 함수 //

Void freenode(NODE\* temp)

Free(temp); 맞나?

// 수건돌리기를 한 번 할 때마다 NODE 1 <-> NODE 2 가 일어나고, count에 변화가 없다. //

// 따라서, swap 함수가 제격이다. //

// NODE 끼리 mynum 이랑 next\* 는 냅두고 STUDENT data 만 swap 한 뒤, data.jo만 서로 swap 해주면 된다. 개꿀 //

// 단, 마지막에는 탈락자가 생기므로 swap 이후 NODE temp2 에 removenode 함수를 적용하여 예외적으로 구성한다. //

Void swapnode(NODE\* temp1, NODE\* temp2);

// 명령어 PLAY를 구현하는 함수 안에서 FILE\* input2 로 game.txt 를 읽는다. //

Void play (TEAM\*\* tlist)

Swapnode (a , b); // a가 b 자리로 가고, b가 a자리로 가서 다시 술래가 된다.

If\* (input == EOF)

Removenode(b);

// 게임 결과를 result\_game.txt 에 저장하는 함수, 1조~5조 오름차순으로 차례대로 저장한다. //

Void savegame(TEAM\*\*)

ㅡㅡㅡㅡㅡㅡㅡㅡ아래의 함수는 굳이 안해도 되는데, assn5.pdf 에서 명령어 별로 함수를 구현하라고 해서 위의 함수들을 조합해서 집어넣으려고 함.ㅡㅡㅡㅡㅡㅡㅡ

// SHOW 함수//

// ADD 함수 //

// REMOVE 함수 //

// QUIT 함수 head부터 pop 를 count 번 실행한다고 생각한다. // Main 에서 for 문을 5번 돌려서

Void quit(TEAM\* 1 2 3 4 5)

Freenode를 count 만큼 실행한다.

Int main( )

{

FILE\* input

Char name, gender, dept;

Int stuid, jo;

STUDENT data;

TEAM\* Tlist[5];

For(int I = 0; I < 5; I++) -> TEAM 1; TEAM 2; TEAM 3; TEAM 4; TEAM 5; 이차원 배열

While(getstu

}

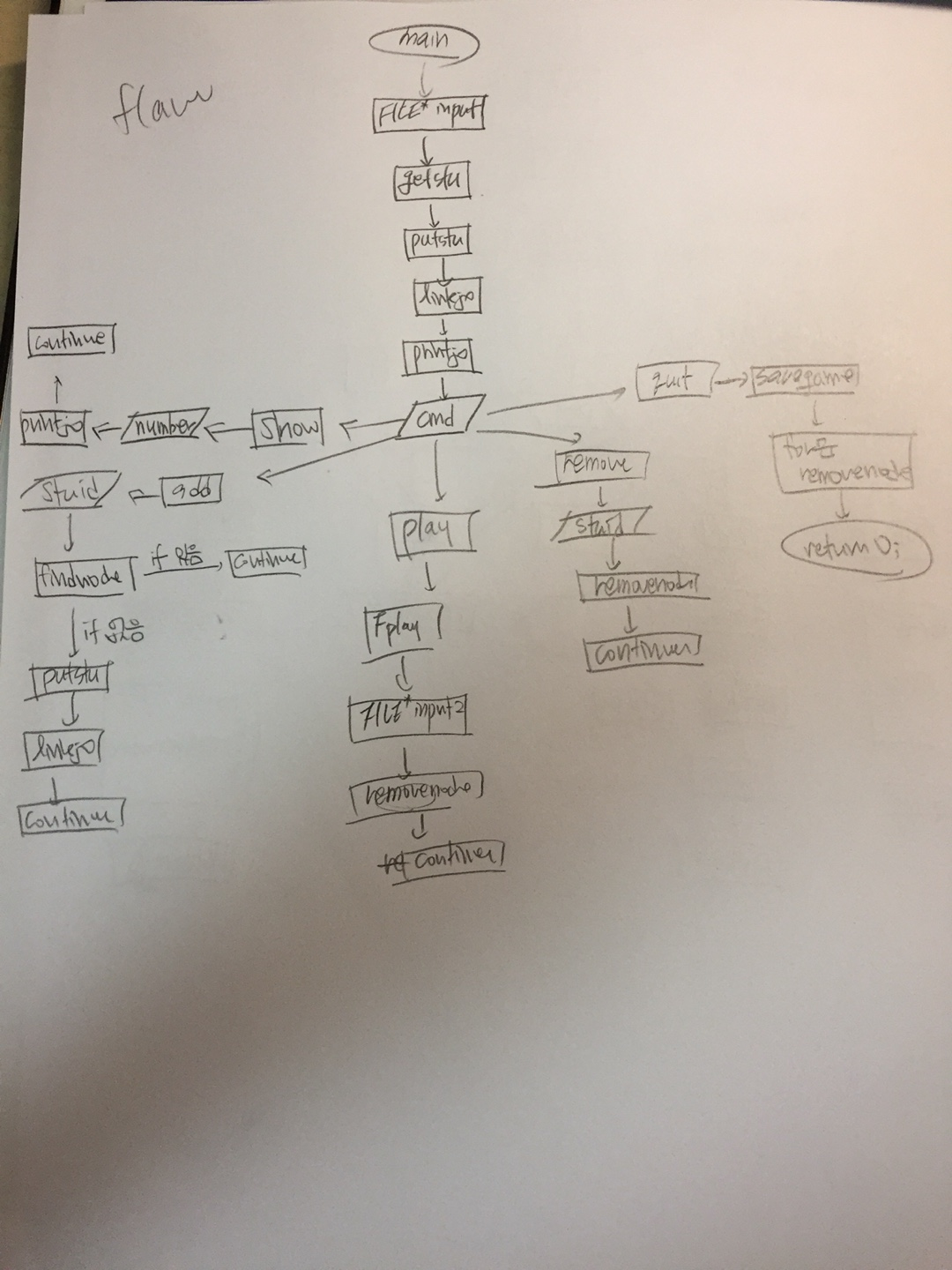
처음에 어싸인을 작성하기 이전에, 워드를 통해 함수 프로토 타입을 구현해놓고 어싸인을 시작했다.

하지만, 어싸인을 직접 작성하는 과정에서 함수의 반환 인자, 함수 인자 가 조금씩 변화되어서, 프로그래밍을 하기 이전에 문제에 대한 고찰이 부족했다는 느낌을 받았고,

이전 assn4 에서는, 프로그래밍 이전에 작성했던 수도코드를 이용해서 함수가 바뀌는 것 없이 그대로 프로그래밍을 할 수 있을 정도로 문제에 대해 깊이 생각해보았는데, 상대적으로 Assn5가 더 어려웠던 것 같다.

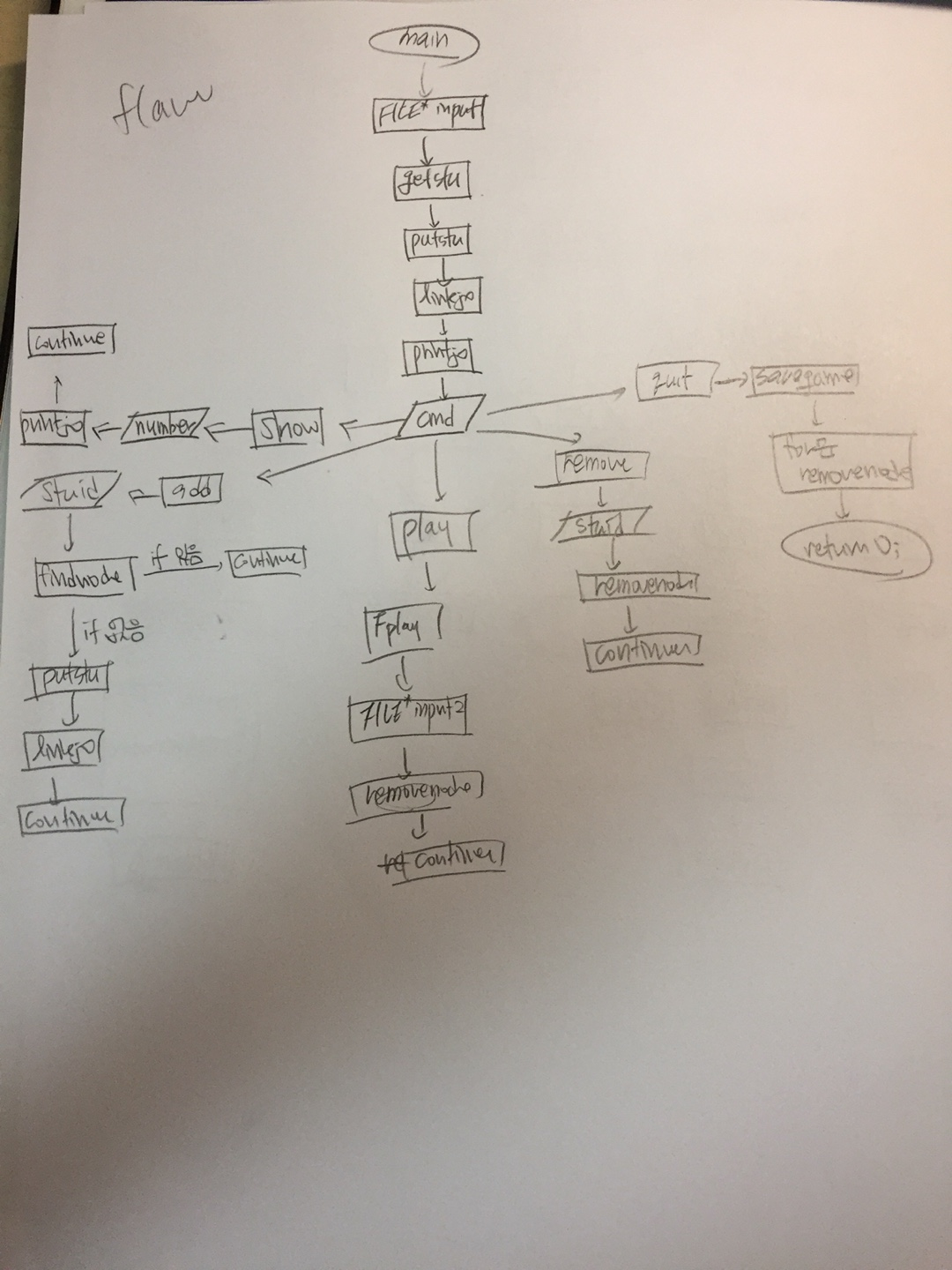
1. 전체구조

<structure chart>

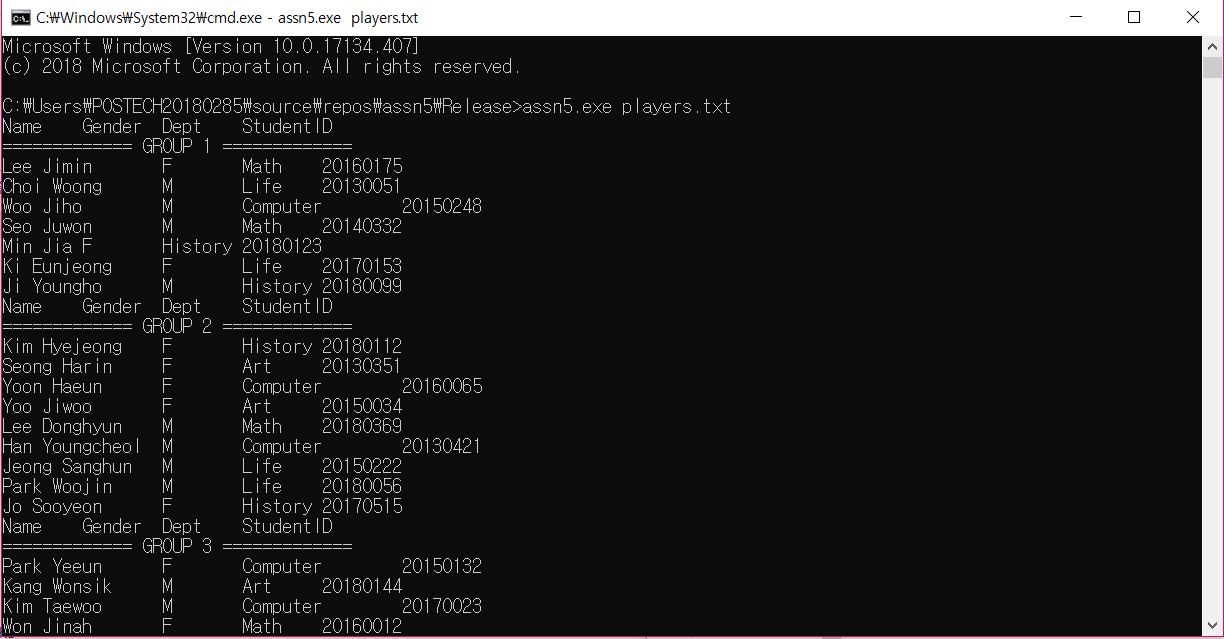


명령어 show, add, play, remove, quit 마다 각각 독립적으로 역할을 하고 있다. 다만 모듈화로 인해 removenode 등등 특정 기능을 하는 함수를 구현했으므로, 그 함수들은 다양한 곳에서 사용되고 있다.

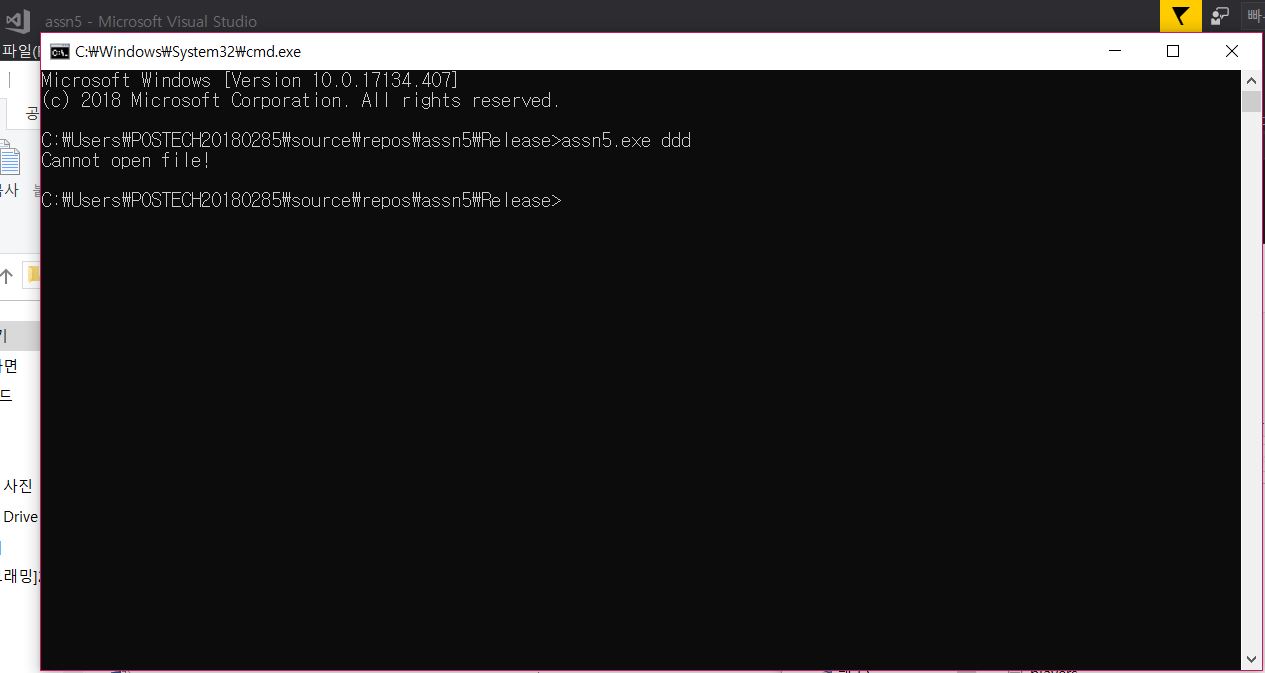
<flow chart>



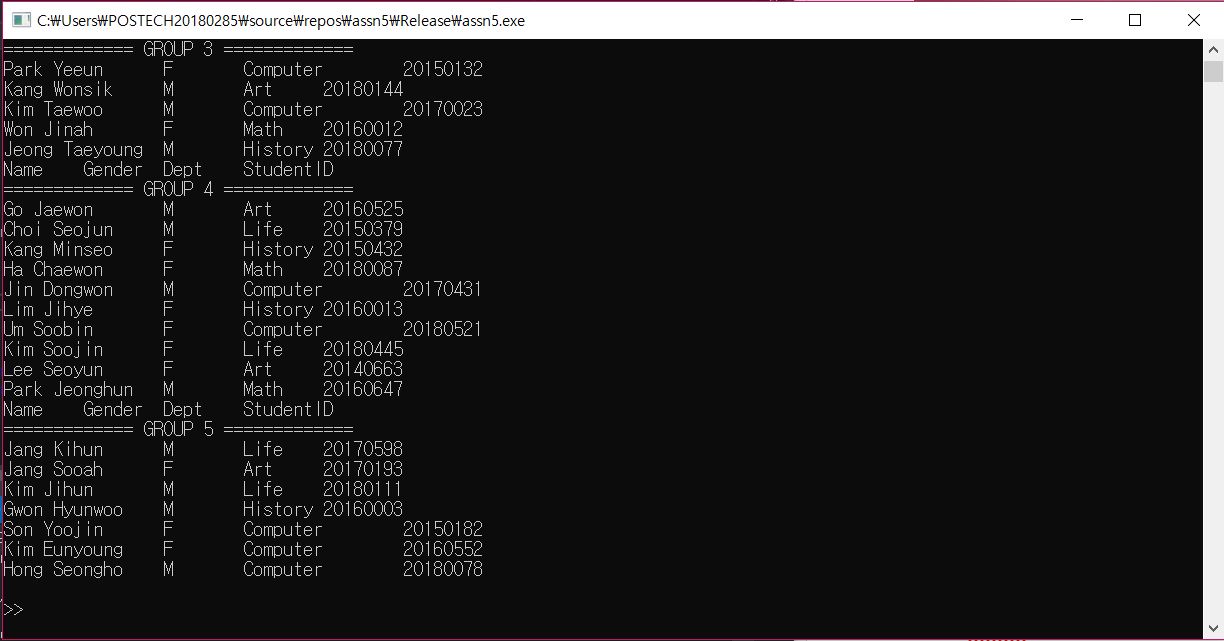
1. 실행예제



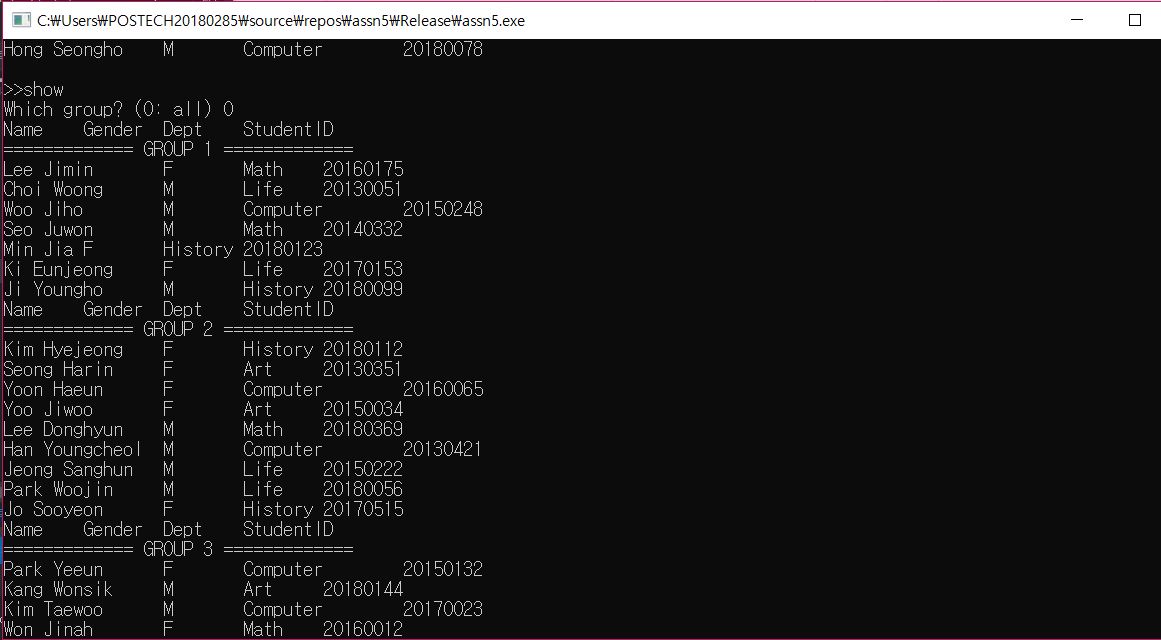
cmd창에서 실행해보았더니 잘 열린다.



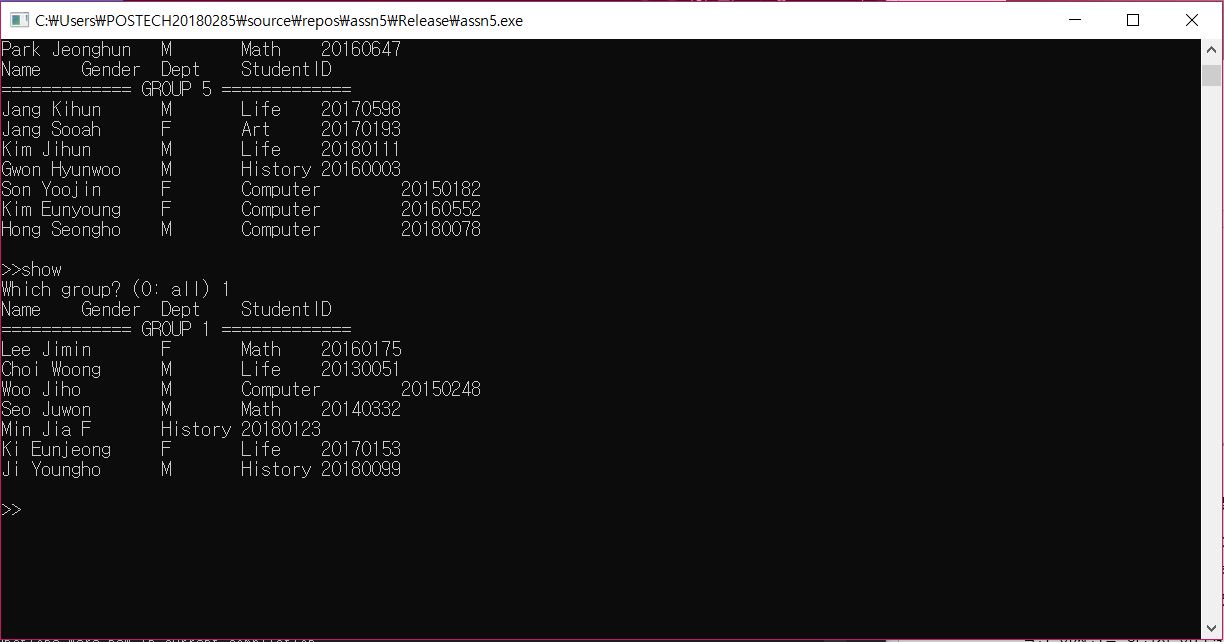
cmd창에서 에러처리도 잘 된다!!



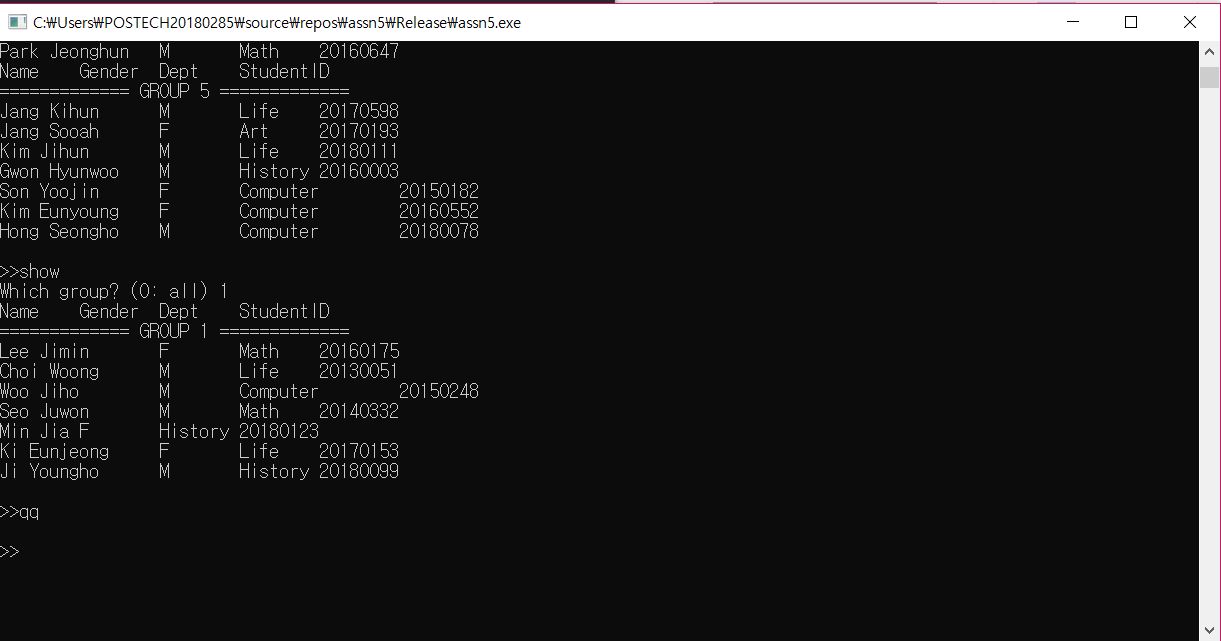
처음에 실행시키면, 자동적으로 1조 ~ 5조까지 printjo를 통해 show 해준다.



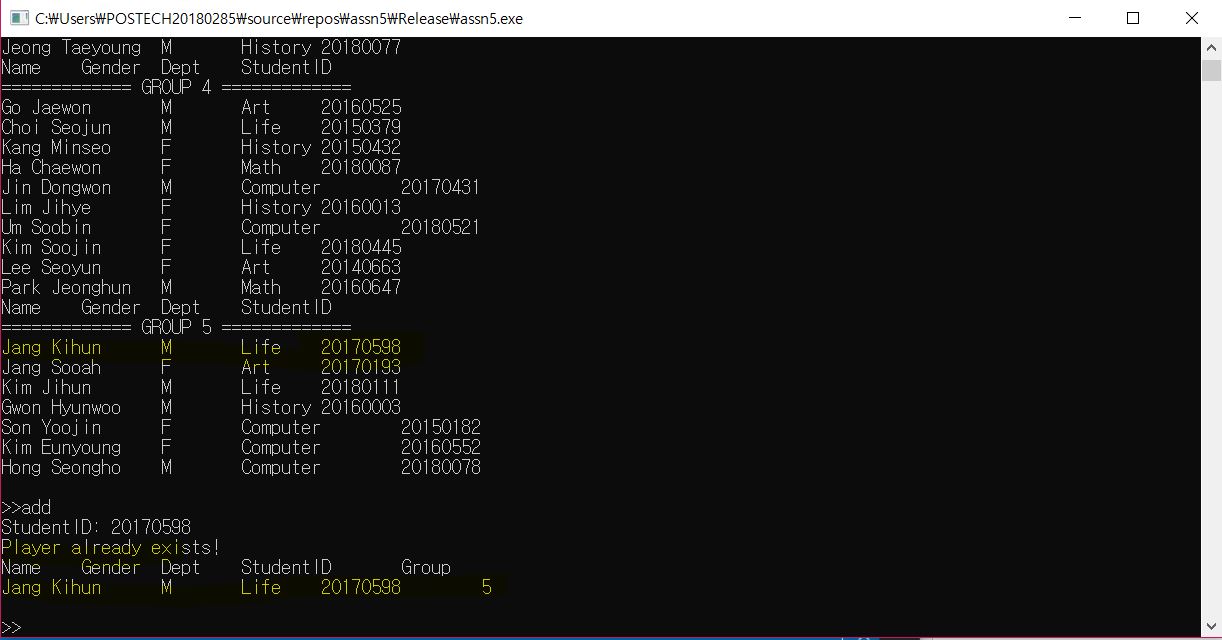
Show를 입력한 뒤, 0을 입력해서 1조 ~ 5조까지 모두 출력시켜 보았다.



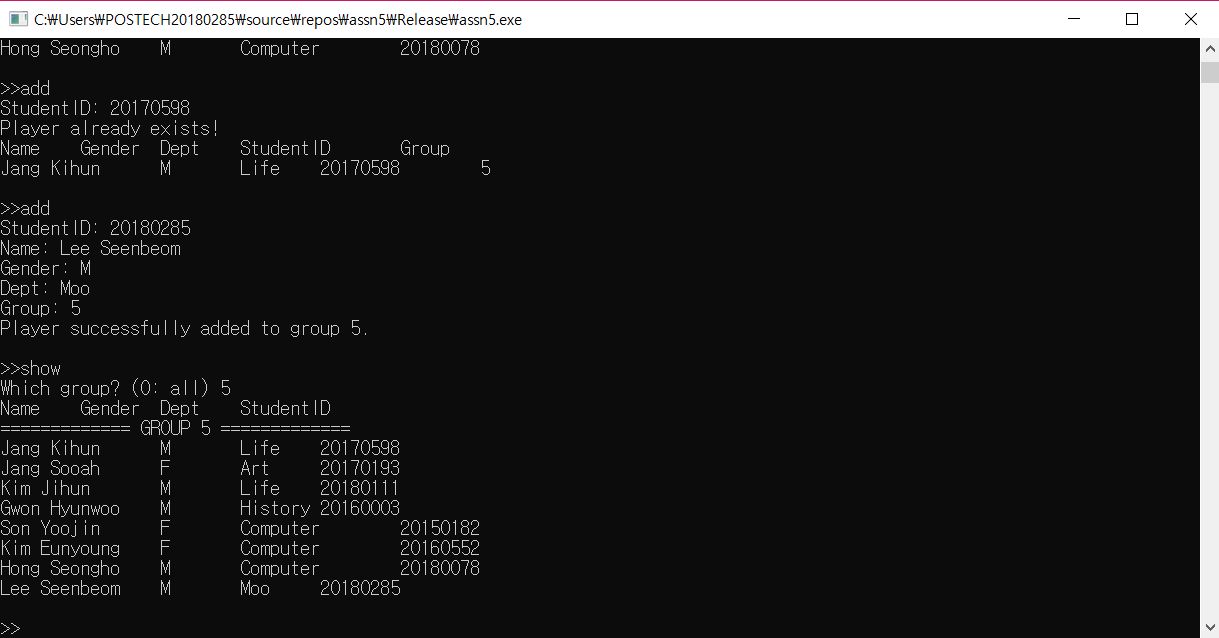
이번에는 show를 입력한 뒤, 1을 입력해서 1조만 출력해보았다.



qq라는 이상한 명령어를 입력하니, 프로그램이 알아서 무시한다.



Add 함수를 하려고 했는데, 20170598라는 학번이 이미 있나보다, 이미 있는 친구라고 알려준 뒤 정보까지 알려준다!

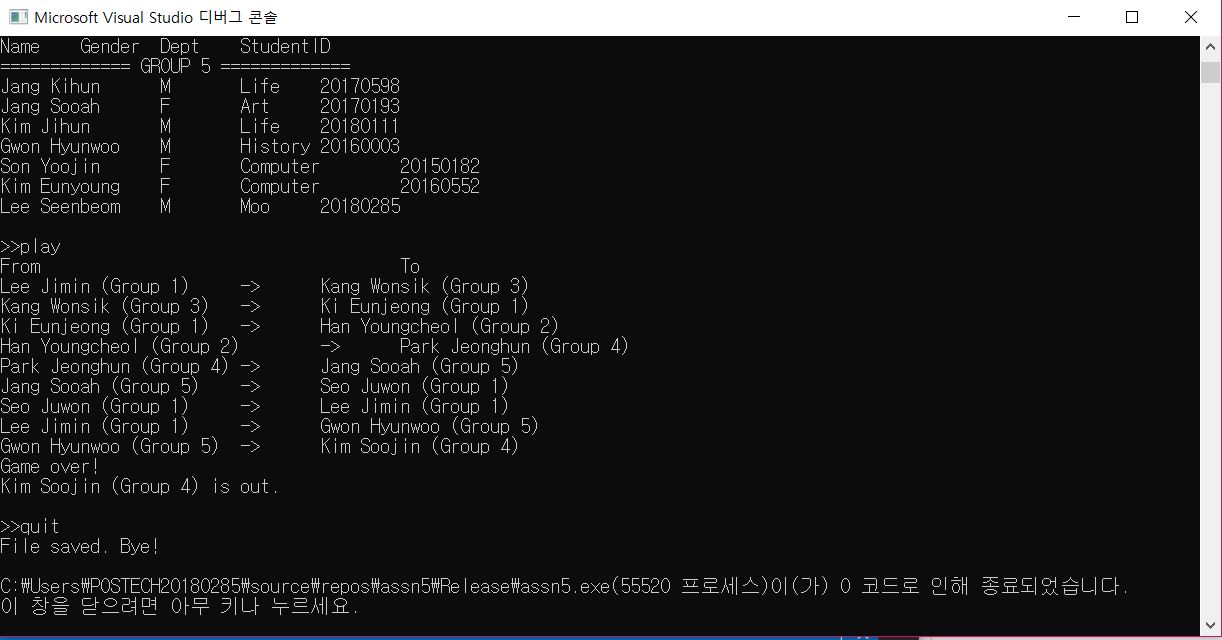


Add 함수를 통해 나를 5조에 추가해보았다.

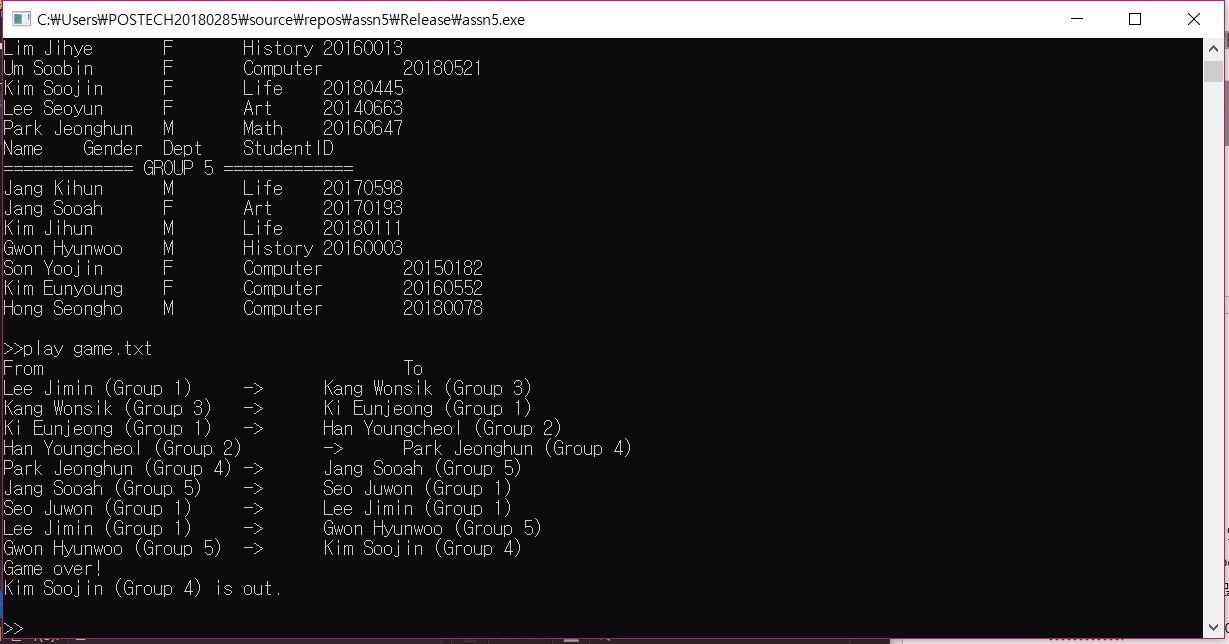


Remove를 통해 친구 한 명을 지우려고 했는데, 처음에는 ‘11’이라는 학번을 입력하니까, 없는 친구라고 나왔다.

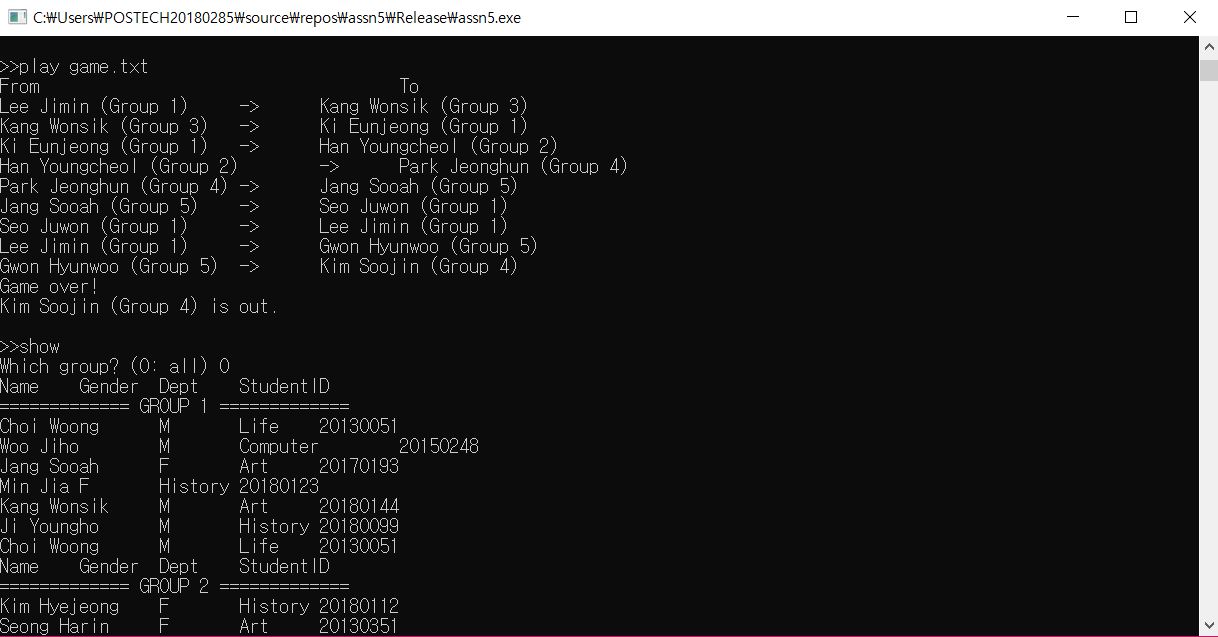
그래서 20180078을 입력했더니, 5조에서 내 위에있던 친구가 사라졌다!



그냥 play를 입력했더니 play가 잘 돌아간다.

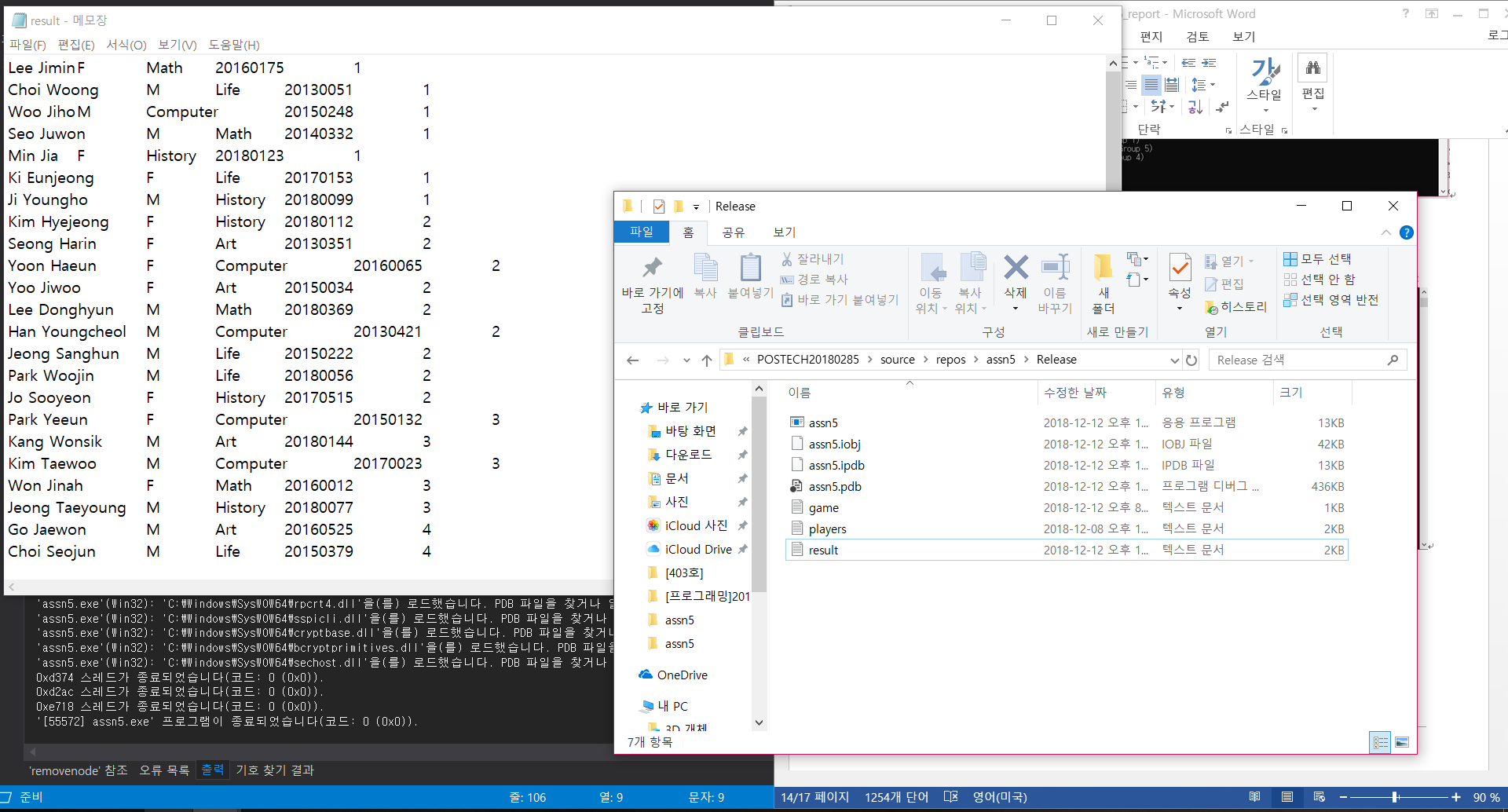


프로그램을 껐다 킨 뒤 play game.txt 를 입력해도 잘 돌아간다.



보면은, play를 통해 장수아라는 사람이 5조 -> 1조로 갔는데,

Play 이후에 장수아라는 사람이 1조에 있는 것을 볼 수 있다. 잘 되었다.



Quit을 입력하니, 결과가 잘 저장되고 프로그램이 free를 모두 한 뒤 종료되었다!

1. 토론

이번 어싸인 기간도 역시,, 신문사 마감일과 학생회에서 전학대회라는 의결기구를 운영하는데, 지난 번에 바빠서 2학기 동안 집을 딱 한 번 갔었다. 그래서 지지난주에 집을 다녀왔고, 지난주에는 주말에 학생회칙 개정을 위한 전학대회가 열려서 거의 하루종일 회의를 했다 ㅜㅠㅠ 주말에 어싸인을 할 시간이 없어서 이번에도 평일에 급하게 하느라 힘들었다.

어려웠던 점을 꼽자면,

1. Play 함수를 잘못 이해해서 어려웠다.

처음에는, 예를들어 1조의 3번 친구가 2조의 2번 친구에게 수건을 준다면, 게임이 종료되기 전까지는 탈락자가 없으므로, 두 NODE->data 만 서로 swap 해서, 속의 STUDENT라는 알맹이만 바꿔주면 되는 줄 알았다.

그래서, swapnode(NODE\* temp1, NODE\* temp2) 라는 함수를 이용해서 play를 구현하려고 했는데, 이렇게 될 경우

만약 첫 술래가 1조의 1번이었다면, 1조의 1번이 누구던지간에, 계속 1조의 1번 자리만 술래가 되는 오류가 일어났다. (즉, 마지막에는 항상 1조의 1번만 지워짐)

결국, 어싸인 pdf 파일 맨 뒤에 ‘오류처리’라고 적혀있는 것처럼, 술래가 된 인원을 해당 TEAM의 링크드 리스트에서 잠시 빼놓은 뒤, 자리를 빼앗을 친구의 NODE에다가 연결시키고, 술래가 된 친구의 NODE는 또다시 TEAM의 링크드 리스트에서 빼놓는 방식을 택했다.

1. 명령어 show Show Show … 를 모두 소문자로 입력하게 하는 것이 string.h에 있는 함수를 몰랐다면 하기 어려웠을 것 같다.

이전에 혼자서 문자열을 복습하다가 인터넷에서 tolower 이라는 함수를 봤었는데, 그것을 이용해서 처리할 수 있었지만, 이 함수를 기억해내지 못했다면 처리하는데 어려움을 느꼈을 것 같다.

1. 이름 사이의 공백을 읽는 것이 어려웠다.

이름 사이의 공백과 \t를 어떻게 구분할지 어려웠다.

결국, 이름 사이의 공백은 %s %s 로 읽은 뒤, strcat 함수를 두 번 사용해서

Name1 뒤에 “ “을 추가하고, 이후 name1 뒤에 name2를 추가해서 이름 문자열을 만들었다.

또한 \t는, fscanf 를 할 때 %s\t%c … 이런 식으로 \t를 넣어주니까 컴퓨터가 알아서 이를 인식한다는 것을 알았다.

1. 결론

링크드 리스트는 단순 배열을 이용한 데이터 관리보다 저장공간을 효율적으로 사용할 수 있다는 장점이 있다.

하지만, 이번 어싸인을 수행하면서 링크스 리스트는 ‘탐색’이 어렵다는 단점을 너무나도 절실히 느꼈다. (특히, 이번에는 단방향 리스트만 사용할 수 있어서 더 어려웠다.)

얘를들어, 내가 지우고 싶은 NODE 가 temp2 였다면, 내가 알고있어야 하는 것은 temp2의 주솟값이 아니라, temp2의 이전 NODE의 주솟값이었다.

그래야지

Temp = pPre -> next

pPre -> next = temp -> next

free(temp);를 통해 링크드 리스트를 이어줄 수 있기 때문이었다.

링크드 리스트는 정말 어려운 것 같다.. 시간이 된다면 더 공부해서 익숙해졌으면 좋겠다.