/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name : 양성민

\* Student ID : 20163124

\* Program ID : hw3

\* Description : data 파일을 사용해서 student의 name, ID, major, grade를 입력 받은 후, 여러가지

\* 함수를 사용해서 조건에 부합하는 student의 data를 출력하는 프로그램

\* Algorithm : 1. data파일에 있는 student의 정보를 받을 구조체 Node를 생성함.

\* 2. class를 선언해주고 class 안에 node의 제일 앞인 head를 포인터변수로 선언하고

\* list가 비어있으므로 head를 0으로 설정함

\* 3. data1을 받을 temp를 node로 선언하고 오름차순으로 정렬하기 위해 p와 q를

\* 선언하여 name을 기준으로 오름차순으로 정렬한다.

\* 4. if (head == 0) { head = temp;}를 사용하여 리스트가 비어있으면 temp를 head로 설정함

\* head가 크다면 head를 temp->next로 넘기고 temp를 head로 저장함 temp가 head보다

\* 크다면 p와 q를 사용해 하나씩 넘어가며 기존 값과 비교하며 temp보다 큰 기존

\* 값이 있으면 그 전에 있는 q에 저장함

\* 5. List가 가지고 있는 현재 student의 data를 모두 출력하기 위해 p를 head로 초기화

\* 하여 next를 통해 하나씩 넘어가며 data들을 찾고 출력함

\* 6. 원하는 grade에 해당하는 student의 data를 출력하기 위해 p를 head로 초기화하고

\* list가 비어있지 않다면 p가 null이 될 때까지 p = p->next 하고 입력된 grade에

\* 해당되는 data를 찾아서 출력함

\* 7. data1과 data2를 저장한 리스트가 하나라도 비어있지 않다면 data1을 저장한

\* 리스트를 head부터 null까지 List ll에 저장하고 data2를 저장한 리스트를 이어서

\* List ll에 저장하여 data1과 data2를 합친 리스트를 생성함

\* 8. 데이터파일 2개를 합친 List에서 CS전공을 출력하기 위해 리스트가 비어있지

\* 않을 때 p를 head로 설정하여 null이 될 때까지 p = p->next하여 major가 CS에

\* 해당하는 student data를 찾아서 출력함

\* (insert를 제외한 모든 함수는 head == 0 일 때 "List Empty"를 출력함)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct Node {

char name;

int ID;

string major;

char grade;

Node \*next;

Node(char value, int num, string word, char abc) {

name = value; //student의 이름

ID = num; //student의 학번

major = word; //student의 전공

grade = abc; //student의 성적

next = 0;

}

};

class List {

private:

Node \*head;

public:

List() { head = 0; }

//student의 정보를 저장할 List 생성

void insertList(char value, int num, string word, char abc);

void searchGrade(char abc); //입력한 성적에 해당되는 student만 출력

void searchMajor(string word); //입력한 전공에 해당되는 student만 출력

void list(); //입력된 student의 정보를 모두 출력

List merge(List a, List b); //data1과 data2를 합쳐서 리스트에 저장

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : insertList

\* description : Node를 입력받고 입력받은 Node의 name을 비교하여 오름차순으로 student의

\* 정보를 저장함

\* Variables : \*temp - 입력받은 값을 temp Node에 저장

\* \*p, \*q - Node 2개를 사용하여 기존 값에서 p가 하나씩 넘어가고 q는 p의 뒤를

\* 따라서 temp의 값을 오름차순으로 sort함

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void List::insertList(char value, int num, string word, char abc) {

Node \*temp = new Node(value, num, word, abc); //생성된 Node를 temp에 저장

Node \*p = head;

Node \*q = 0;

if (head == 0) { //head가 null이면 temp값을 head로 저장

head = temp;

}

else if (temp->name < head->name) { //입력된 값을 오름차순으로 sort함

temp->next = head;

head = temp;

}

else {

while ((p != 0) && (p->name < temp->name)) {

q = p;

p = p->next;

}

if (p != 0) {

temp->next = p;

q->next = temp;

}

else

q->next = temp;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : searchGrade

\* description : 원하는 성적을 입력하여 입력된 성적에 해당하는 student의 정보만 출력

\* Variables : \*p - head부터 하나씩 넘어가며 입력된 성적에 해당되는 student를 찾음

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void List::searchGrade(char abc) {

Node \*p;

if (head != 0) { //리스트가 비어있지 않고

p = head;

while (p != 0) { //p = 0이 아니면

if (p->grade == abc) { //원하는 성적에 해당하는 student 모두 출력

cout << p->name << " ";

cout << p->ID << " ";

cout << p->major << " ";

cout << p->grade << endl;

}

p = p->next;

}

cout << endl;

}

else

cout << "List Empty" << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : searchMajor

\* description : 원하는 전공을 입력하여 입력된 전공에 해당하는 student의 정보만 출력

\* Variables : \*p - head부터 하나씩 넘어가며 입력된 전공에 해당되는 student를 찾음

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void List::searchMajor(string word) {

Node \*p;

if (head != 0) { //리스트가 비어있지 않고

p = head;

while (p != 0) { //p = 0이 아니면

if (p->major == word) { //원하는 전공에 해당하는 student 모두 출력

cout << p->name << " ";

cout << p->ID << " ";

cout << p->major << " ";

cout << p->grade << endl;

}

p = p->next;

}

}

else

cout << "List empty" << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* function : list

\* description : 리스트에 저장된 모든 student의 정보를 출력

\* Variables : \*p - head부터 하나씩 넘어가며 출력해야 하는 정보를 찾음

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void List::list() {

Node \*p;

if (head != 0) { //리스트가 비어있지 않다면

p = head;

while (p != 0) { //저장된 정보 모두 출력

cout << p->name << " ";

cout << p->ID << " ";

cout << p->major << " ";

cout << p->grade << endl;

p = p->next;

}

cout << endl;

}

else

cout << "List empty" << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* funtion : merge

\* description : 받은 데이터파일 2개를 합쳐서 새로운 리스트에 저장함

\* Variables : ll - 데이터파일 2개를 합친 정보를 저장할 리스트

\* \*p - data1의 정보를 head부터 하나씩 넘어가며 찾음

\* \*q - data2의 정보를 head부터 하나씩 넘어가며 찾음

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List List::merge(List a, List b) { //데이터 파일 2개를 합쳐서 List ll에 저장

List ll;

Node \*p = a.head;

Node \*q = b.head;

if (a.head != 0 || b.head != 0) { //List a나 b가 비어있지 않다면

while (p != 0) { //p가 null이 될때까지 data를 받음

ll.insertList(p->name, p->ID, p->major, p->grade);

p = p->next;

}

while (q != 0) {//q가 null이 될때까지 data를 받음

ll.insertList(q->name, q->ID, q->major, q->grade);

q = q->next;

}

}

else

cout << "List empty." << endl;

return ll;

}

int main() {

List A; //data1을 받을 리스트

List B; //data2를 받을 리스트

List C; //A와 B를 합쳐서 저장할 리스트

ifstream fin;

fin.open("data1.txt", ios::in);

for (int i = 0; i < 7; i++) { //data1을 순서대로 각각의 자료형으로 A에 받음

char a; int b; string c; char d;

fin >> a >> b >> c >> d;

A.insertList(a, b, c, d);

}

fin.close();

fin.open("data2.txt", ios::in);

for (int j = 0; j < 3; j++) { //data2를 순서대로 각각의 자료형으로 B에 받음

char a; int b; string c; char d;

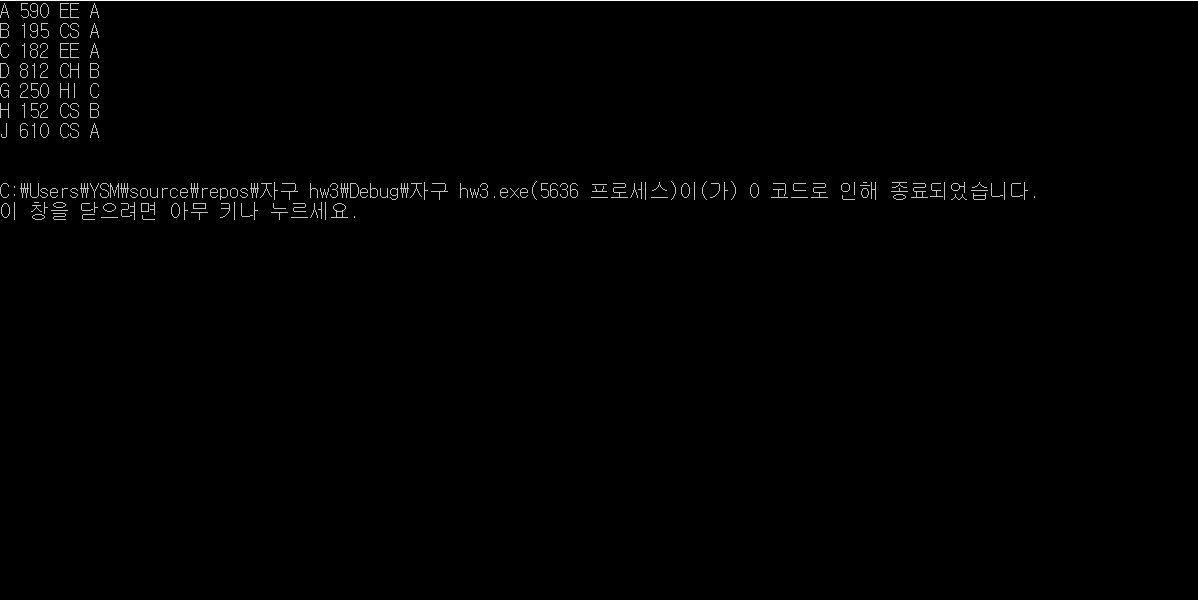
fin >> a >> b >> c >> d;

B.insertList(a, b, c, d);

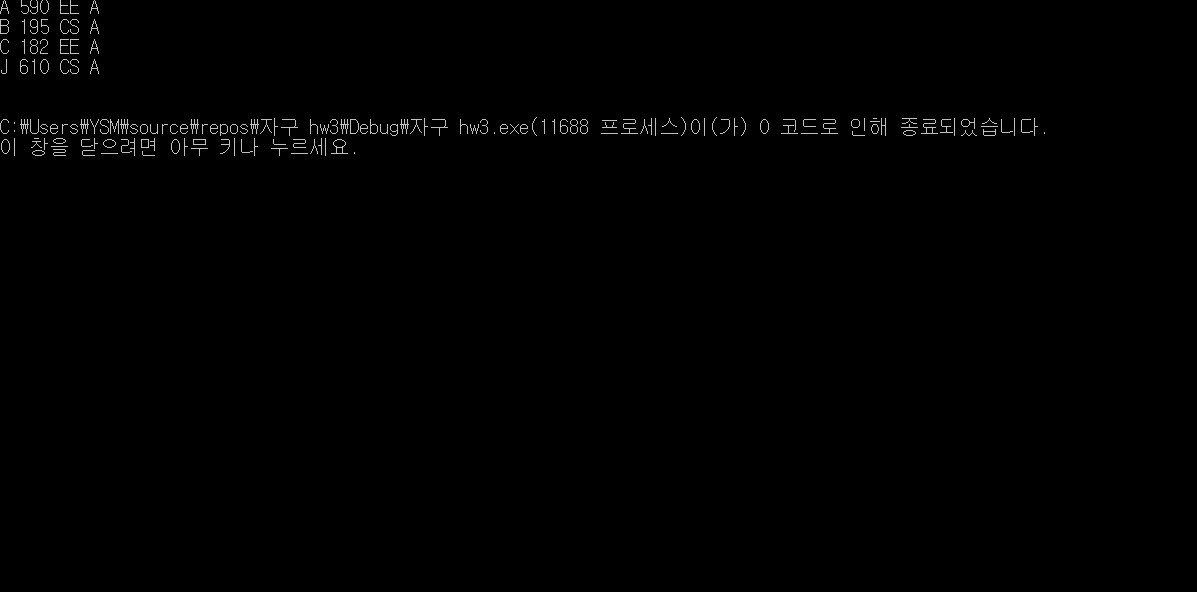
}

//명령 실행

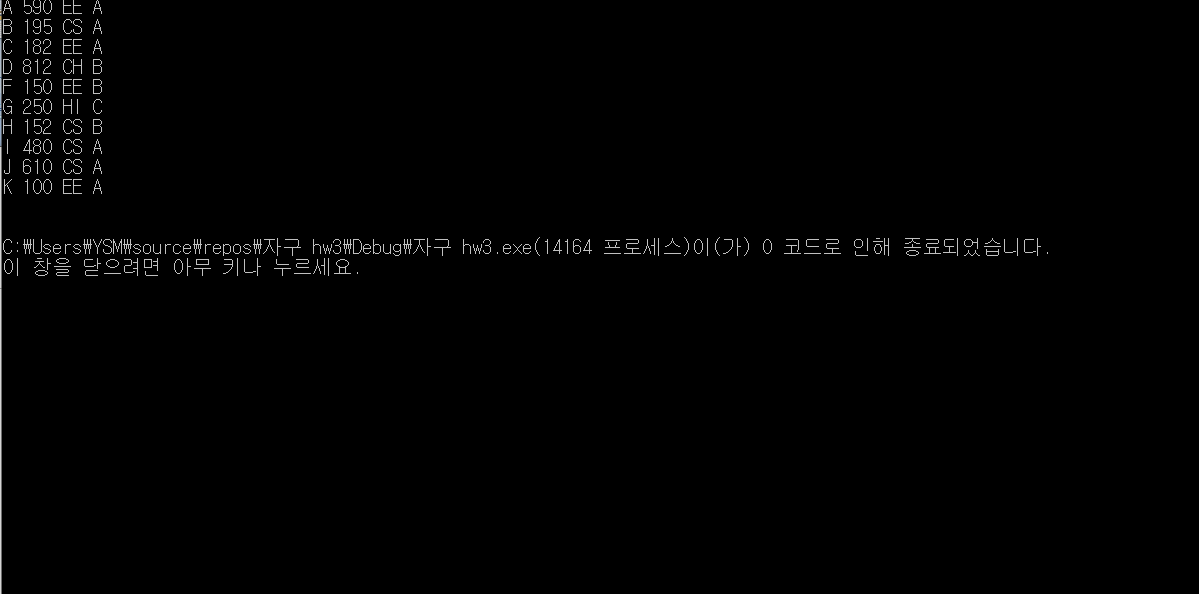
A.list(); //정보를 출력



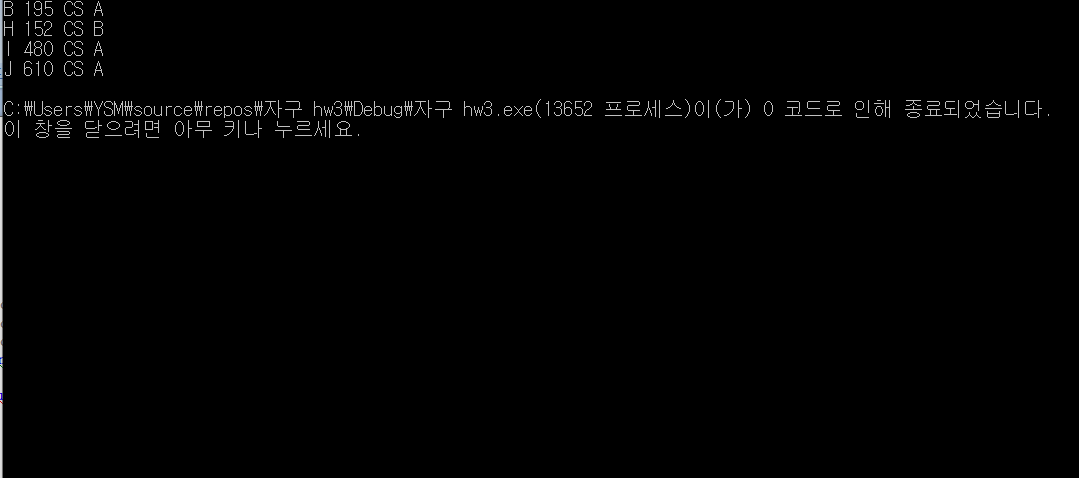
A.searchGrade('A'); //A 성적 출력



C = C.merge(A, B); //A리스트와 B리스트를 합쳐서 C리스트에 받음



C.searchMajor("CS"); //C리스트에서 CS전공을 출력



}