```
*********************************************************************************************
* Name : 차윤성
* Student Id : 20163162
* Program Id : Hw#3
* Description :
* - 학생의 이름, Id, 전공, 성적이 있는 data파일을 불러와 Linked List를 이용하여 name을 기준으로 오름차
*   순으로 정리한다.
* - 저장된 data로부터 성적 또는 전공별로 출력될 수 있게 한다,
* - 이미 불러온 data파일이 아닌 새로운 data파일을 읽고 새로운 Linked List를 이용하여 기존 data list와 새
*   로운 data list를 병합한 후에, 찾으려는 전공에 해당하는 학생의 data를 출력한다.
* Algorithm :
* - 먼저 학생의 name, Id, major, grade를 받아서 저장할 구조체 Node를 만든다. 기존 Node는 data 하나였
*   지만, 받아와야 하는 data의 개수가 4개이므로 Node 내에서 data를 4개 만들어준다. Node내에는 다음
*   node를 저장할 포인터 변수 next도 같이 선언해준다.
* - Linked List를 구현할 class를 선언해준다. class 내부에는 List의 제일 앞 node를 의미하는 포인터변수
*   head를 선언해주고, 최초 instance를 생성할 시에 list 내에는 아직 아무런 data가 없으므로 head를 0으로
*   설정해준다.
* - 모든 검사에 대해서 head==0 일 경우는 list가 비어있다는 의미이다. 그러므로 isEmpty 함수를 만들어 준
*     다.
* - data를 List에 삽입 시에, 추가하려는 node temp를 선언해주고, 비교 node p와 q를 선언해준다. 이 때 삽
*   입되는 각 node들은 name에 대하여 오름차순으로 정렬한다.
* - List가 empty일 경우 temp가 가장 첫번째 node이므로 head를 temp로 초기화해준다.
* - temp name의 ASCII 코드 값이 head name의 ASCII 코드 값보다 작으면 head의 앞에 배치가 되어야 하
*   므로 temp 의 포인터변수 next를 head가 되어주도록 하고, temp를 새로운 head로 설정해준다.
* - temp name값이 head name값 보다 크다면 node p를 head로 초기화해준다. 그 후 p가 List의 끝이 되거
*   나 p의 name값이 temp의 name값 보다 커질 때 까지 반복하며 p를 다음 node로 옮겨주고 q를 p가 옮겨
*   가기 전 node로 설정해준다.
* - 반복이 끝난 후 p가 0일 경우, 즉 p가 List내 node의 마지막 다음일 경우 temp를 맨 마지막에 삽입해야한
*   다. 그러므로 q의 next를 temp로 설정해주고, temp의 next는 0으로 초기화되어있기 때문에 그대로 둔다.
* - p가 0이 아니라면 p와 q 사이에 temp node가 삽입되어야 하므로 q의 next를 temp로 설정해주고 temp의
*   next를 p로 설정하여 모든 node가 연결되도록 한다.
* - 현재 List내의 모든 data를 display 하려면 p를 head로 초기화 해준 후, p가 0이기 전까지 data를 출력한
*   후 p의 next를 p로 설정해주어 다음 node로 옮겨간다.
* - 특정 data값을 찾으려 할 때 List가 비어있지 않으면 p를 head로 설정해준다.
* - p==0이기 전까지, 즉 List 내 node의 마지막까지 반복하며 각 node별 조건에 만족하는 major 또는 grade
*   값을가지는 node의 name, Id, major, grade를 출력한 후 다음 노드로 넘어가기 위해 node p 자체를 다
*   음 node로 바꿔준다.
* - 두 개의 List를 병합하는 함수 mergeList는 return type을 List로 설정해준다.
* - 매개변수로 받은 두 개의 List를 병합하여 저장시킬 List c를 선언하고, 각각의 List의 head를 저장해 줄
*   node p와 q를 선언해준다.
* - 두 개의 List가 하나라도 비어있지 않을 경우 각각의 List들에 대해 비교 node가 0이 아닐 때 까지 반복하
*   며 List c에 모든 node를 insert한다. 그 후 비교 node를 각 List의 다음 node로 옮겨준다.
* - c에 대한 insert가 종료되면 List c를 return 한다.
* - insert를 제외한 모든 함수에 대하여 isEmpty가 true일 경우에는 “List is empty”를 출력한 후 함수를 종
*   료한다.
* - data1.txt는 List llA에 저장하고, data2.txt는 List llB에 저장, 두개의 List를 병합한 List는 llC에 저장한
*   후 llA를 display하고, llA에서 성적이 A인 학생들을 출력하고, llC에서 전공이 CS인 학생들을 출력한다.
* Variables :
* struct Node : List안에 들어갈 구조체 이름
* name : data에서 받아온 학생의 이름을 저장해주는 변수
* Id : data에서 받아온 학생의 Id를 저장해주는 변수
* major : data에서 받아온 학생의 전공을 저장해주는 변수
* grade : data에서 받아온 학생의 점수를 저장해주는 변수
* next : linked list에서 각 Node가 다음으로 연결되는 Node를 가리키는 포인터 변수
* class List : List를 클래스로 지정해
* Node *head : 각 List별 가장 앞 Node를 가리키는 포인터 변수
* bool isEmpty() : List가 비어있는지 검사하는 함수
* void insertList() : List에 node를 삽입해주는 함수
```

```
* void displayList() : List내의 node를 모두 출력해주는 함수
* void searchMajor() : List내 특정 전공값을 가진 node를 찾아주는 함수
* void searchGrade() : List내 특정 성적값을 가진 node를 찾아주는 함수
* List mergeList() : 매개변수로 받은 2개의 List를 하나의 List로 병합해주는 함수. retrun type은 List.
*********************************************************************************************
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

struct Node {   // 다뤄야 하는 data를 Node안에 선언해주고 next를 만들어 줌
        char name;
        int Id;
        string major;
        char grade;
        Node *next;
        Node(char value, int num,string str,char chr) { // 구조체 Node 생성자 함수
                name = value;  // 매개변수로 받은 value를 name으로 설정
                Id = num;              // 매개변수로 받은 num을 Id로 설정
                major = str;    // 매개변수로 받은 str을 major로 설정
                grade = chr;    // 매개변수로 받은 chr를 grade로 설정
                next = 0;              // next를 0으로 초기화
        }
};
class List {
private:
        Node *head = 0;
public:
        List() {
                head = 0;
        }       // 처음 List를 만들 때 head를 0으로 설정해 줌.
        bool isEmpty();
        void insertList(char name,int Id,string major,char grade);
        void displayList();
        void searchMajor(string major);
        void searchGrade(char grade);
        List mergeList(List a, List b);
};
```

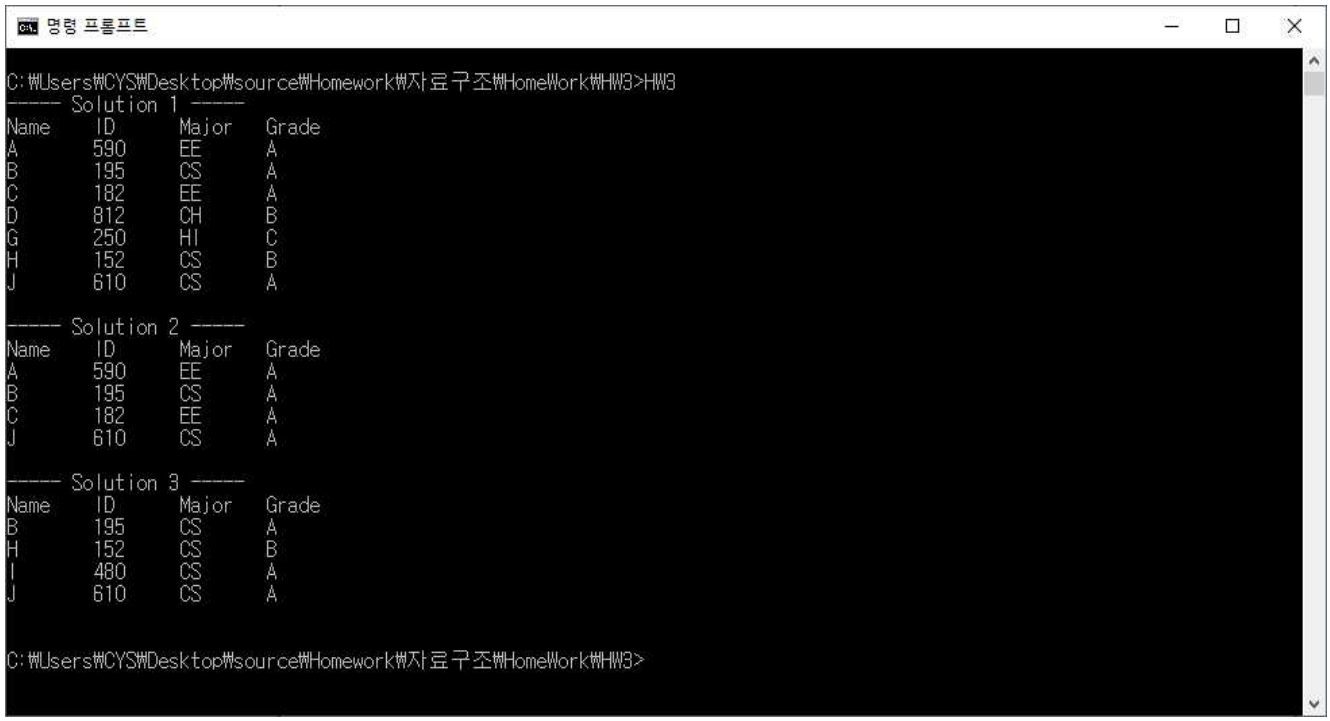
```
*****************************************************************************************
* function : searchMajor
* description : searchMajor의 사용 목적은 특정 전공 값을 가진 Node를 찾아 data를 출력하는 것이다.
* Variable :
* Node *p : 각 Node를 검사하기 위한 임시 Node 포인터 변수
* cnt : 찾으려는 major가 있는지 없는지 판별하기 위한 int형 변수
*****************************************************************************************
void List::searchMajor(string major) {
	Node *p;
	int cnt = 0;	// cnt가 0이면 찾는 major가 없다는 의미
	if (!isEmpty()) {
		p = head;
		while (p != 0) {
			if (p->major == major) {
				cout << "name: " << p->name << " ";
				cout << "Id: " << p->Id << " ";
				cout << "Major: " << p->major << " ";
				cout << "Grade: " << p->grade << endl;
				cnt++;
			}
			p = p->next;	// 찾은 후 다음 Node로 넘어감
		}
		if (!cnt) cout << major << " is not in the list." << endl;
	}
	else {
		cout << "List is empty." << endl;
	}
}
*****************************************************************************************
* function : searchGrade()
* description : searchGrade의 사용목적은 특정 성적 값을 가진 Node를 찾아 data를 출력하는 것이다.
* Variable :
* Node *p : 각 Node를 검사하기 위한 임시 Node 포인터 변수
* cnt : 찾으려는 grade가 있는지 없는지 판별하기 위한 int형 변수
*****************************************************************************************
void List::searchGrade(char grade) {
	Node *p;
	int cnt = 0;
	if (!isEmpty()) {
		p = head;
		while (p != 0) {
			if (p->grade == grade) {
				cout << "name: " << p->name << " ";
				cout << "Id: " << p->Id << " ";
				cout << "Major: " << p->major << " ";
				cout << "Grade: " << p->grade << endl;
				cnt++;
			}
			p = p->next;
		}
		if (!cnt) cout << grade << " is not in the list." << endl;
	}
	else {
		cout << "List is empty." << endl;
	}
}
```

```
*********************************************************************************************
* function : isEmpty()
* description : isEmpty() 사용목적은 List가 비어있는지 아닌지 검사하는 것이다.
*********************************************************************************************
bool List::isEmpty() {
        if (head == 0) return true;       // if head = 0 then list is empty
        else return false;
}
*********************************************************************************************
* function : insertList()
* description : insertList()의 사용목적은 List내에 새로운 node data를 추가하는 것이다.
* Variables :
* Node *temp : 새로 추가하려는 data를 저장하는 node
* Node *p : 새로 추가될 node의 위치 지정을 위해 사용되는 임시 Node 포인터 변수
* Node *p : 새로 추가될 node의 위치 지정을 위해 사용되는 임시 Node 포인터 변수
*********************************************************************************************
void List::insertList(char name, int Id, string major, char grade) {
        Node *temp = new Node(name,Id,major,grade);
        //추가하려는 List의 데이터 값들을 받아와서 새로운 Node를 만듬.
        Node *p = head;
        Node *q = 0;

        if (isEmpty()) {
                head = temp;
        }
        else if (temp->name < head->name) {
                // name이 char형이므로 ASCII코드값을 비교
                temp->next = head;      // head보다 값이 작으면
                head = temp;            // 추가하려는 List가 head가 됨
        }
        else{
                p = head;
                while ((p != 0) && p->name < temp->name) {
                        q = p;          // 추가하려는 List의 name이 더 작고
                        p = p->next;    // head가 0이 아닐 때 까지 검사
                }
                if (p != 0) {           // 자리를 찾으면 p와 q의 사이에 List 추가
                        temp->next = p;
                        q->next = temp;
                }
                else                    // p가 0이 되면 List의 마지막에 추가
                        q->next = temp;
        }
}
```

```
*****************************************************************************************
* function : displayList()
* description : displayList()의 사용목적은 현재 List안에 들어있는 모든 node들에 대해 data를 출력하는 것이다.
* Variable :
* Node *p : 각 Node를 지정하기 위한 임시 Node 포인터 변수
*****************************************************************************************
void List::displayList() {
        Node *p;
        if (!isEmpty()) {
                p = head;
                while (p) {       // Node p가 0이 아닐때 까지 List 출력
                        cout << "name: " << p->name << " ";
                        cout << "Id: " << p->Id << " ";
                        cout << "Major: " << p->major << " ";
                        cout << "Grade: " << p->grade << endl;
                        p = p->next;
                }
        }
        else
                cout << "List is empty." << endl;
}
*****************************************************************************************
* function : mergeList()
* description : mergeList()의 사용목적은 매개변수로 받은 두 개의 List를 한 개의 List로 병합하는 것이다.
* Variable :
* List c : 두 개의 List를 합쳐서 저장하기 위한 List
* Node *p : 매개변수 List a의 node를 저장하기 위한 포인터 변수
* Node *q : 매개변수 List b의 node를 저장하기 위한 포인터 변수
*****************************************************************************************
List List :: mergeList(List a, List b) {
        List c;           // return type이 List임
        Node *p;
        Node *q;
        p = a.head;
        q = b.head;
        if (!a.isEmpty() || !b.isEmpty()) { // 2개의 List가 하나라도 비어있지 않으면
                while (p != 0) {
                        c.insertList(p->name, p->Id, p->major, p->grade);
                        p = p->next;    // List c에 매개변수로 받은 List a를 insert
                }
                while (q != 0) {
                        c.insertList(q->name, q->Id, q->major, q->grade);
                        q = q->next;    // List c에 매개변수로 받은 List b를 insert
                }
        }
        else cout << "List is empty." << endl;   // 2개의 List가 모두 비어있으면
        return c;
}
```

```
int main() {
	List llA; //data1를 저장할 List llA 선언
	List llB; //data2를 저장할 List llB 선언
	List llC; //data1과 data2를 저장할 List llC 선언

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

	if (infile.fail()) {
		cout << "can't open the input file." << endl;
		exit(1);
	}
	for (int i = 0; i < 7; i++) {
		char a;
		int b;
		string c;
		char d;
		infile >> a >> b >> c >> d;
		llA.insertList(a, b, c, d);
	}

	infile.close();	//data1.txt를 닫음

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

	if (infile.fail()) {
		cout << "can't open the input file." << endl;
		exit(1);
	}
	for (int i = 0; i < 3; i++) {
		char a;
		int b;
		string c;
		char d;
		infile >> a >> b >> c >> d;
		llB.insertList(a, b, c, d);
	}
	cout << "Solution 1" << endl;
	llA.displayList();
	cout << endl;

	cout << "Solution 2" << endl;
	llA.searchGrade('A');
	cout << endl;

	cout << "Solution 3" << endl;
	llC = llC.mergeList(llA, llB);
	llC.searchMajor("CS");
}
```

****
* Output
*


명령 프롬프트
C:₩Users₩CYS₩Desktop₩source₩Homework₩자료구조₩HomeWork₩HW3>HW3
----- Solution 1 -----
Name ID Major Grade
A 590 EE A
B 195 CS A
C 182 EE A
D 812 CH B
G 250 HI C
H 152 CS B
J 610 CS A
----- Solution 2 -----
Name ID Major Grade
A 590 EE A
B 195 CS A
C 182 EE A
J 610 CS A
----- Solution 3 -----
Name ID Major Grade
B 195 CS A
H 152 CS B
I 480 CS A
J 610 CS A
C:₩Users₩CYS₩Desktop₩source₩Homework₩자료구조₩HomeWork₩HW3>

*
****