

# 자료구조 과제 보고서

[학생정보 관리 프로그램 설계]

12141579 윤찬미

010-9287-1679

[cmmcme0604@gmail.com](mailto:cmmcme0604@gmail.com)

## I. 개요

- 설계의 목적

- 학생 정보 관리 프로그램을 인접 리스트로 구현된 그래프 이용하여 설계한다.

- 요구 사항

- 초기에 파일에 저장된 학생들의 정보를 입력받아 그래프에 입력 할 수 있다.
- 초기에 파일에 저장된 두 학생 사이의 도로 정보를 그래프에 입력 할 수 있다.
- 새로운 학생을 추가 할 수 있다.
- 새로운 도로를 추가 할 수 있다.
- 학생, 도로의 정보를 출력 할 수 있다.
- 두 학생이 인접한지, 학생과 도로가 연결되어 있는지 확인 할 수 있다.

- 개발 환경

- Microsoft Visual Studio 2013
- 언어 : C++

## II. 자료구조 및 기능

- Graph을 구현하기 위한 자료구조

### 1. Vertex

```
typedef struct Vertex    //정점의 구조체
{
    int num;           //학번
    string Email;      //이메일
    int dev;           //차수
    struct ConEdge *Con;    //연결되어있는 간선의 시퀀스
    Vertex() : num(0), dev(0), Con(NULL) {};    //초기화해준다
}Vertex;
```

- 구조체를 이용하여 Vertex를 생성해 주었다.
- Vertex 구조체 안에 학번, 이메일, 차수(연결된 간선의 수), 다음 정점 연결된 간선을 가리키는 포인터형 Edge를 선언한 후 각 값을 초기화 시켰다.

## 2. VertexList

```
typedef struct VertexList //각 정점을 가리키고 있는 Vertex 시퀀스를 구조체로 정의한다.
{
    VertexList* next; //다음 시퀀스를 가리킴
    struct Vertex* MyVer; //정점을 가리킨다.

    VertexList() : next(NULL), MyVer(NULL) {}; //각 값을 초기화한다.
} VertexList;
```

- 구조체를 이용하여 VertexList를 만들어 주었다.
- 다음 위치를 알려줄 수 있는 Linked-list 구조이다.

## 3. Edge

```
typedef struct Edge
{
    int StreetNum; //도로 번호
    ConEdge* ConEd1; //한쪽 정점에 연결된 간선 시퀀스
    ConEdge* ConEd2; //반대쪽 정점에 연결된 간선 시퀀스
    EdgeList* MyOrder; //엣지 리스트에서 나의 위치
    Vertex *Ver1; //한쪽 정점의 위치
    Vertex *Ver2; //반대쪽 정점의 위치
    Edge() : StreetNum(0), ConEd1(NULL), ConEd2(NULL), MyOrder(NULL), Ver1(NULL), Ver2(NULL)
    {}; //각 값을 초기화한다.
}Edge;
```

- 구조체를 이용하여 Edge를 생성해 주었다.
- Edge 구조체 안에 도로 번호, 다음 간선(도로), 양 끝에 연결된 정점1, 2를 생성해 준 후 초기화 시켰다.

## 4. EdgeList

```
typedef struct EdgeList //그래프에서 간선의 시퀀스
{
    EdgeList* next; //다음 간선
    struct Edge *SelfEd; //가지고 있는 간선의 위치
    EdgeList() : next(NULL), SelfEd(NULL) {}; //초기화

} EdgeList;
```

- 구조체를 이용하여 EdgeList를 만들어 주었다.
- 다음 위치를 알려줄 수 있는 Linked-list 구조이다.

## 5. ConEdge

```
typedef struct Vertex    //정점의 구조체
{
    int num;        //학번
    string Email;    //이메일
    int dev;        //차수
    struct ConEdge *Con;    //연결되어있는 간선의 시퀀스
    Vertex() : num(0), dev(0), Con(NULL) {};    //초기화해준다
}Vertex;
```

– 구조체를 이용하여 Vertex에 연결되어 있는 간선들의 시퀀스이다.

## 6. Graph

```
typedef struct Graph    //그래프 구조체
{
    EdgeList* EDList;    //엣지 리스트 시퀀스
    VertexList* VerList;    //버텍스 시퀀스
    Graph() : EDList(NULL), VerList(NULL) {};    //초기화해줌
}Graph;
```

- 구조체를 이용하여 Graph를 생성해 주었다.
- Graph 구조체 안에 정점과 간선을 가리키는 포인터형 구조체를 선언하고 초기화 시켜 주었다.

### • 프로그램의 기능

#### 1-1. Add\_FileVertex

#### 1-2. void AddVertex(int Num, String EM);

→ Num이라는 학번과 EM이라는 이메일을 가지고 있는 정점을 그래프에 삽입한다. 그래프에 있는 Vertex sequence의 가장 마지막에 새로운 Vertex를 추가 해 준다.

## 2-1. Add\_FileEdge

## 2-2. void AddEdge(int streetnum, int Snum, int Vnum);

-> SNum과 Vnum이라는 학번을 가지고 있는 두 정점 사이를 잇는 간선을 삽입 하는 함수이다.  
두 학번의 위치를 찾은 후, 각 학번이 가지고 있는 연결된 간선의 끝에 해당 간선을 삽입 한다.

## 3. Vertex\* SearchVer (int Num);

-> Num 학번을 가진 정점의 위치를 찾는다. 그래프에서 저장되어있는 정점의 sequence를 따라가며 저장된 학번이 Num과 일치하면 그 위치를 반환해준다.

## 4. Edge\* SearchEd (int Num);

-> Num 학번을 가진 간선의 위치를 찾는다. 그래프에서 저장되어있는 간선의 sequence를 따라가며 저장된 도로번호가 Num과 일치하면 그 위치를 반환해준다.

## 5. void DelVertex (int Num);

-> Num의 학번을 가지고 있는 정점을 찾은 후, 정점에 연결된 간선을 보는데,  
간선이 비어져 있지 않다면 간선의 반대쪽 끝에 있는 정점에 연결된 간선도 삭제 한 후, 다음 간선에  
서도 이 과정을 반복한다. 연결된 간선을 모두 보고 반복문을 종료하고, 정점을 삭제한다.

## 6. void DelEdge(int Num);

-> Num의 도로번호를 가진 간선의 위치를 찾는다. 간선이 존재한다면 간선의 양 쪽 끝에 있는  
정점의 차수를 1 감소 시키고, 각 정점에 연결되어 있는 간선을 찾아 삭제한다.

## 7-1. void printVer(int Num);

## 7-2. void printAlled(int Num);

## 7-3. void printEd(int Num);

## 7-4. void printEd2(int stnum, int Num);

-> 해당 Num을 가지고 있는 Vertex/Edge를 Search 함수를 이용하여 찾은 후 정보를 출력한다.

## 8. void IsAdjacency(int n1,int n2)

-> 해당 n1/n2를 가지고 있는 Vertex를 Search 함수를 이용하여 찾은 후 두 정점 중 차수가  
작은 정점의 연결된 Edge sequence를 따라가며 두 정점이 인접한지 확인한다.

### III.기능별 알고리즘 명세

1-1. Algorithm Add\_FileVertex

1-2. Algorithm AddVertex(int Num, String EM);

//filedata 입력과 새로운 학생 입력의 알고리즘이 유사하기에 1개만 정의

	시간복잡도
Vertex* ver <- new Vertex	1
ver->stunum <- Num;	1
ver->email <- Em;	1
Vertex* _Ver = G->Vertices	1
if _Ver = NULL then	1
G->Vertices <- ver	1
else then	
while ( ¬_Ver->next = NULL)	n
_Ver <- _Ver->next	
_Ver->next <- ver	1

---

$O(n)$

2-1. Algorithm Add\_FileEdge

2-2. Algorithm AddEdge(int streetnum, int Snum, int Vnum);

	시간복잡도
Edge* Ed <- new Edge	1
EdgeList* newEdList <- new EdgeList	1
SV <- SearchVer(Snum)	n
VV<- SearchVer(Vnum)	n
if SV = NULL    VV = NULL then	1
print(Not found)	1
else then	
Ed->street <-streetnum	1
Edge* _Edg <- G->Ed	1
Edge* SE <- SV->Incident	1
Edge* VE <- VV->Incident	1
if _Edg = NULL then	1

G->Ed <- Ed	1
else	
while ( ¬_Edg->next = NULL)	m
_Egd <- _Edg->next	1
_Edg->next <- Edg	1
while ( ¬ SE = NULL)	deg(SV)
SE <- SE -> next	1
while ( ¬ VE = NULL)	deg(VV)
VE <- VE -> next	1

---

$O(m+n)$

### 3. Algorithm SearchVer (int Num);

시간복잡도

VertexList *SearchNow <- Gra->VerList;	1
while (true)	n
if SearchNow = NULL then	
break;	
if SearchNow->MyVer->num = num then	
break;	
else then	
SearchNow <- SearchNow->next	
if SearchNow != NULL then	1
return SearchNow->MyVer;	1
else then	
return NULL;	1

---

$O(n)$

#### 4. Algorithm SearchEd (int Num);

시간복잡도

EdgeList *nowList <- Gra->EDList;	1
while (true)	m
if nowList = NULL then	
break;	
if nowList->SelfEd->StreetNum =StreetNum then	
break;	
else	
nowList <- nowList->next;	
if nowList != NULL then	1
return nowList->SelfEd;	1
else then	1
return NULL;	1

---

0(m)

#### 5. Algorithm DelVertex (int Num);

시간복잡도

---

0(m)

#### 6. Algorithm DelEdge (int Num);

시간복잡도

SearchEdge(streetnum);	m
EdgeList* nowList <- Gra->EDList;	1
if(nowList->SelfEd ==delEdge) then	1
delete nowList	1
else	
while(nowList->next->SelfEd !=delEdge)	m
nowList <- nowList->next	1
EdgeList* delList <- nowList -> next	1



nowList -> next = nowList -> next-> next	1
delete delList	1
repeat Con	m

---

$O(m)$

7-1. Algorithm printVer(int Num);

7-2. Algorithm printAlled(int Num);

	시간복잡도
Vertex* ver=SearchVertex(Num)	n
if(ver = NULL)	1
print(Not found)	1
else	
print(PrintNow->Email,PrintNow->dev)	1

---

$O(n)$

8. Algorithm printEd(int Num);

	시간복잡도
Edge* Ed=SearchEdge(Num)	m
if(Ed = NULL)	1
print(Not found)	1
else	
print(Ed->V1->num,Ed->V2->num)	1

---

$O(m)$

## 9. Algorithm printEd2(int stnum, int Num);

	시간복잡도
Vertex* SerVE = SearchVertex(num)	n
Edge* SerEd = SearchEdge(StreetNum)	m
if (SerVE = NULL    SerEd = NULL) then	1
print(Not available)	1
else	
print(SerEd->V1->num    SerEd->V2->num)	1

---

$O(n+m)$

## 10. Algorithm IsAdjacency (int n1,int n2);

	시간복잡도
Vertex* V1 = SearchVertex(n1);	n
Vertex* V2 = SearchVertex(n2);	n
if (V1 = NULL    V2 = NULL)	1
print(Not found)	
return;	
if (V1->dev = 0    V2->dev = 0)	1
print(Not found)	1
return;	1
if (V1->dev < V2->dev)	1
ConEdge *nowCon <- V1->Con;	1
while (true)	
if (nowCon = NULL)	
print(Not found)	1
break;	1
if (nowCon->MyEd->Ver1/Ver2 = V2)	1
print( nowCon->MyEd->StreetNum)	1
break;	1
else	
nowCon <- nowCon->next;	1
else	
repeat that	

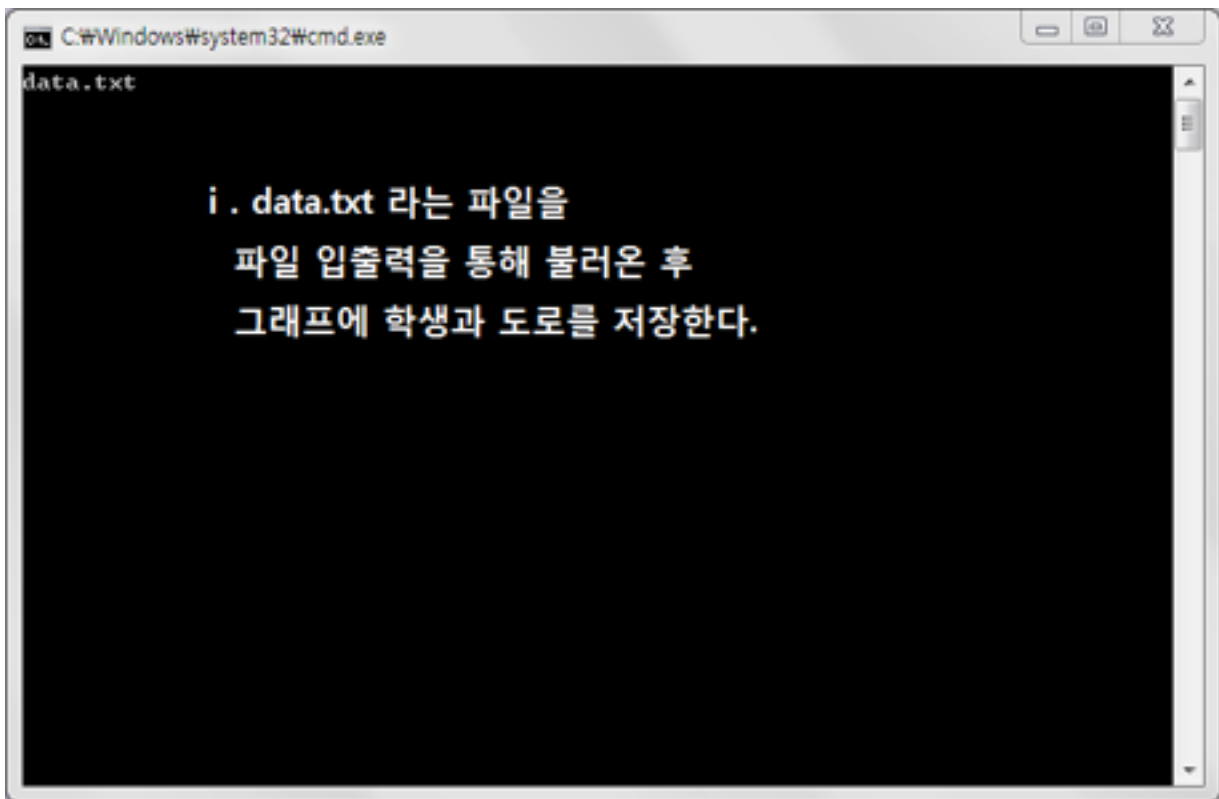
---

$O(n)$

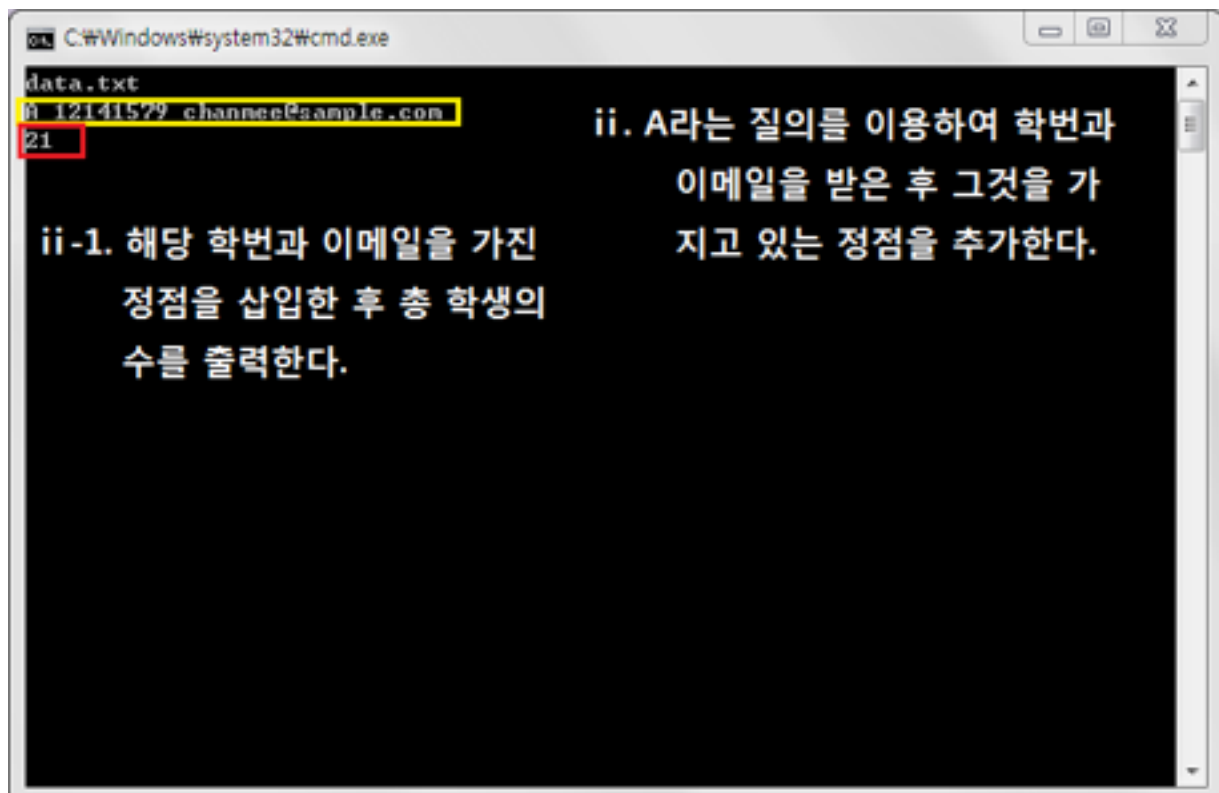
## IV. 인터페이스 및 사용법

### － 실행화면 스크린샷과 기능 설명

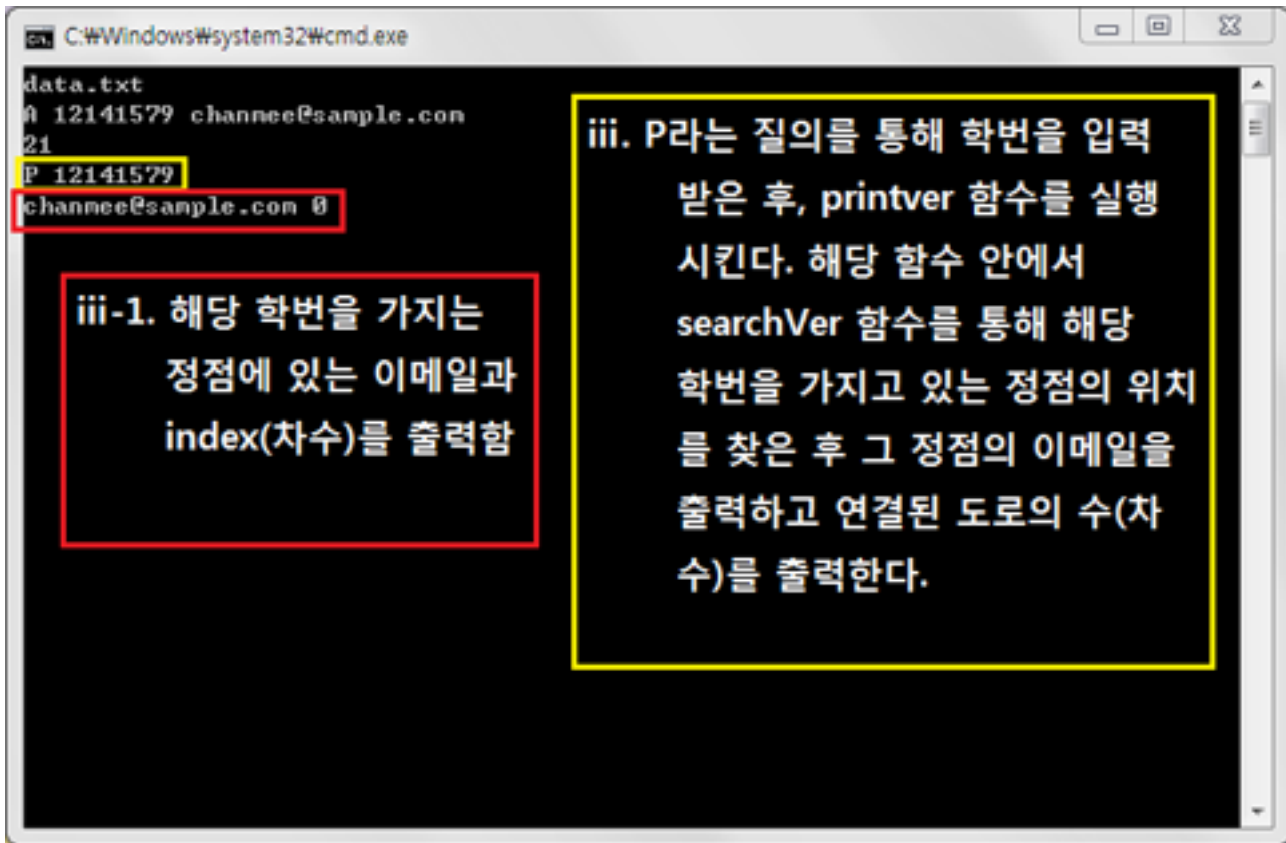
#### 1) 파일 입 출력을 통한 파일 불러오기



#### 2) 학생 추가 질의



### 3) 학생 정보 출력



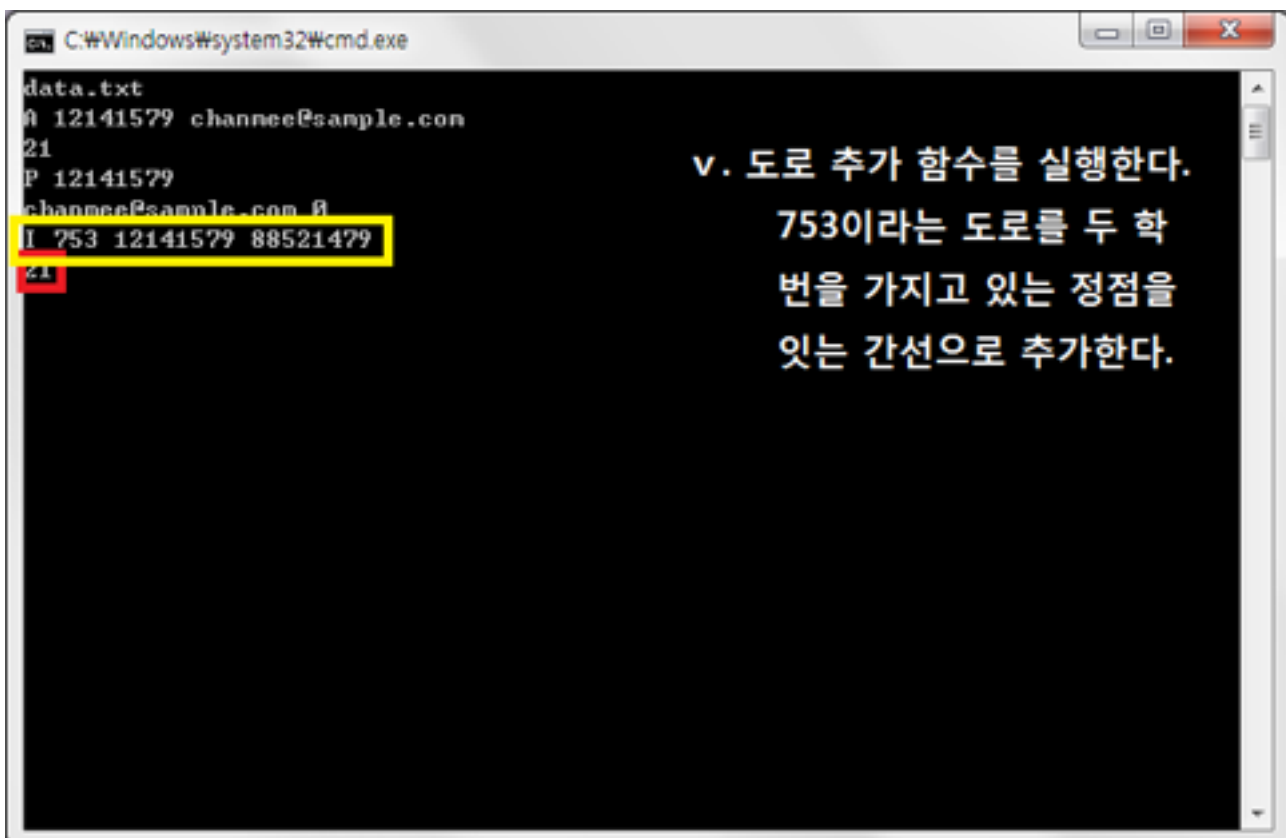
The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The command prompt displays the following text:

```
data.txt
A 12141579 channee@sample.com
21
P 12141579
channee@sample.com 0
```

Two text boxes provide additional context:

- iii-1. 해당 학번을 가지는 정점에 있는 이메일과 index(차수)를 출력함** (iii-1. Output the email and index (degree) of the point corresponding to the given student ID.)
- iii. P라는 질의를 통해 학번을 입력 받은 후, printver 함수를 실행시킨다. 해당 함수 안에서 searchVer 함수를 통해 해당 학번을 가지고 있는 정점의 위치를 찾은 후 그 정점의 이메일을 출력하고 연결된 도로의 수(차수)를 출력한다.** (iii. After inputting the student ID through the query P, execute the printver function. Inside this function, use the searchVer function to find the position of the point corresponding to the given student ID, then output the email of that point and the number of connected roads (degree).)

### 4) 도로 추가



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The command prompt displays the following text:

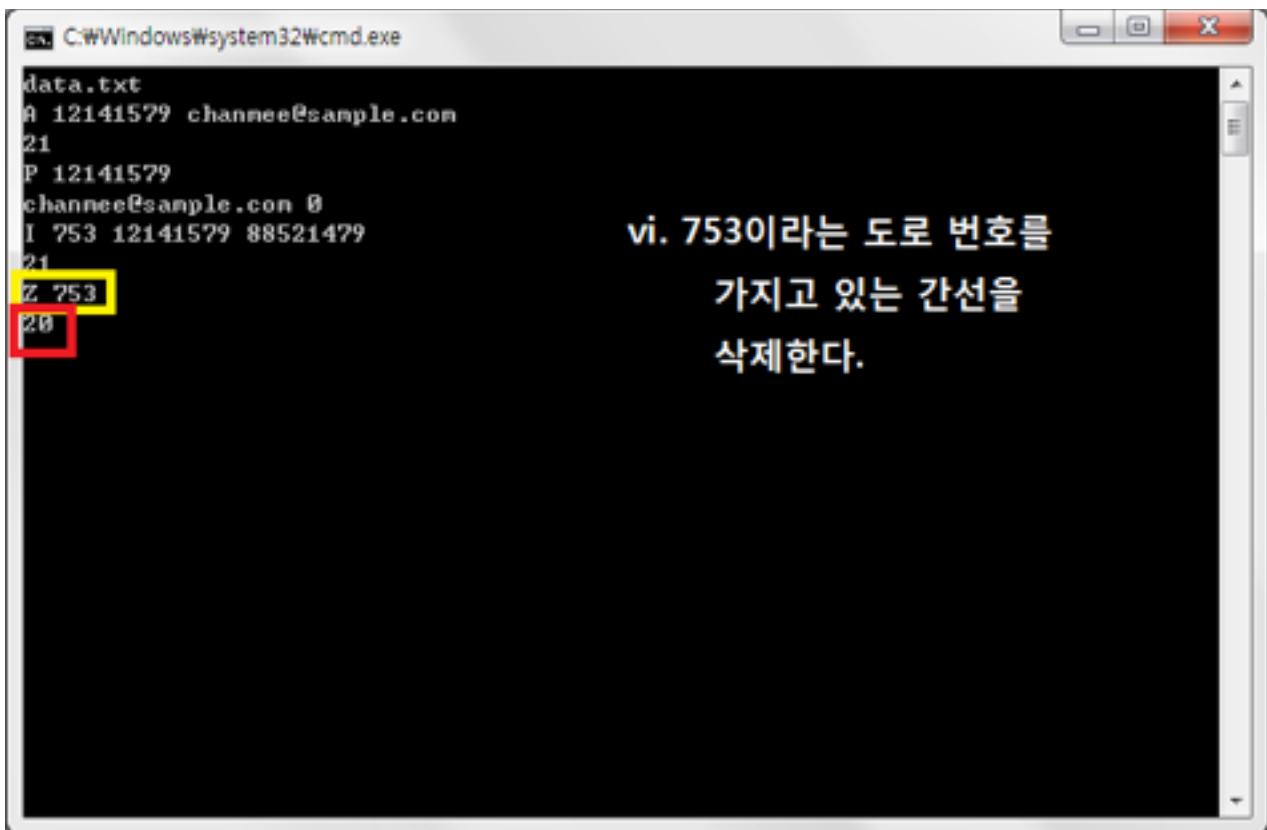
```
data.txt
A 12141579 channee@sample.com
21
P 12141579
channee@sample.com 0
I 753 12141579 88521479
21
```

A text box provides additional context:

- v. 도로 추가 함수를 실행한다. 753이라는 도로를 두 학번을 가지고 있는 정점을 잇는 간선으로 추가한다.** (v. Execute the road addition function. Add the road 753 as an edge connecting the two points corresponding to the given student IDs.)

5) 학생 삭제

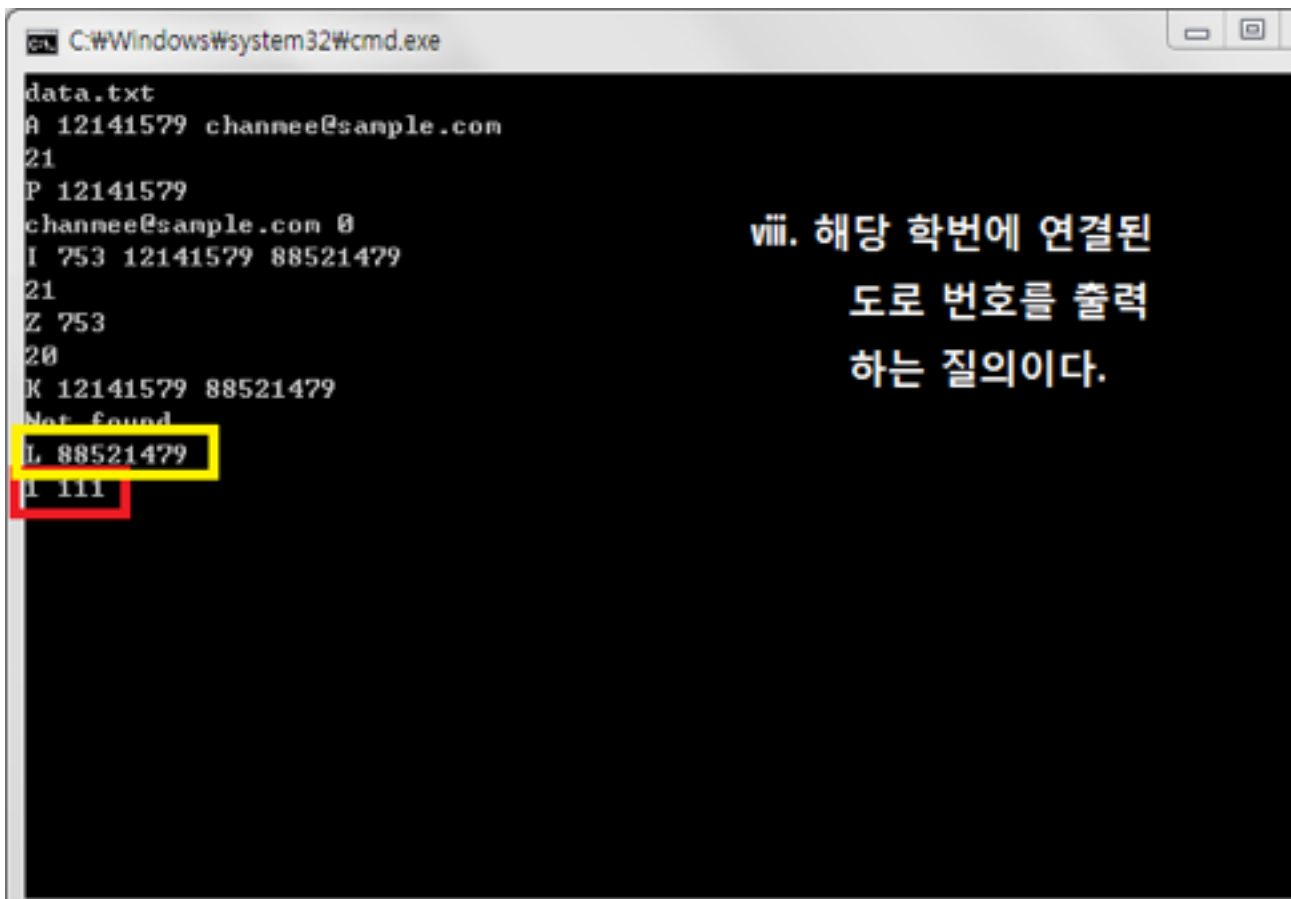
6) 도로 삭제



```
C:\Windows\system32\cmd.exe
data.txt
A 12141579 channee@sample.com
21
P 12141579
channee@sample.com 0
I 753 12141579 88521479
21
Z 753
20
```

vi. 753이라는 도로 번호를  
가지고 있는 간선을  
삭제한다.

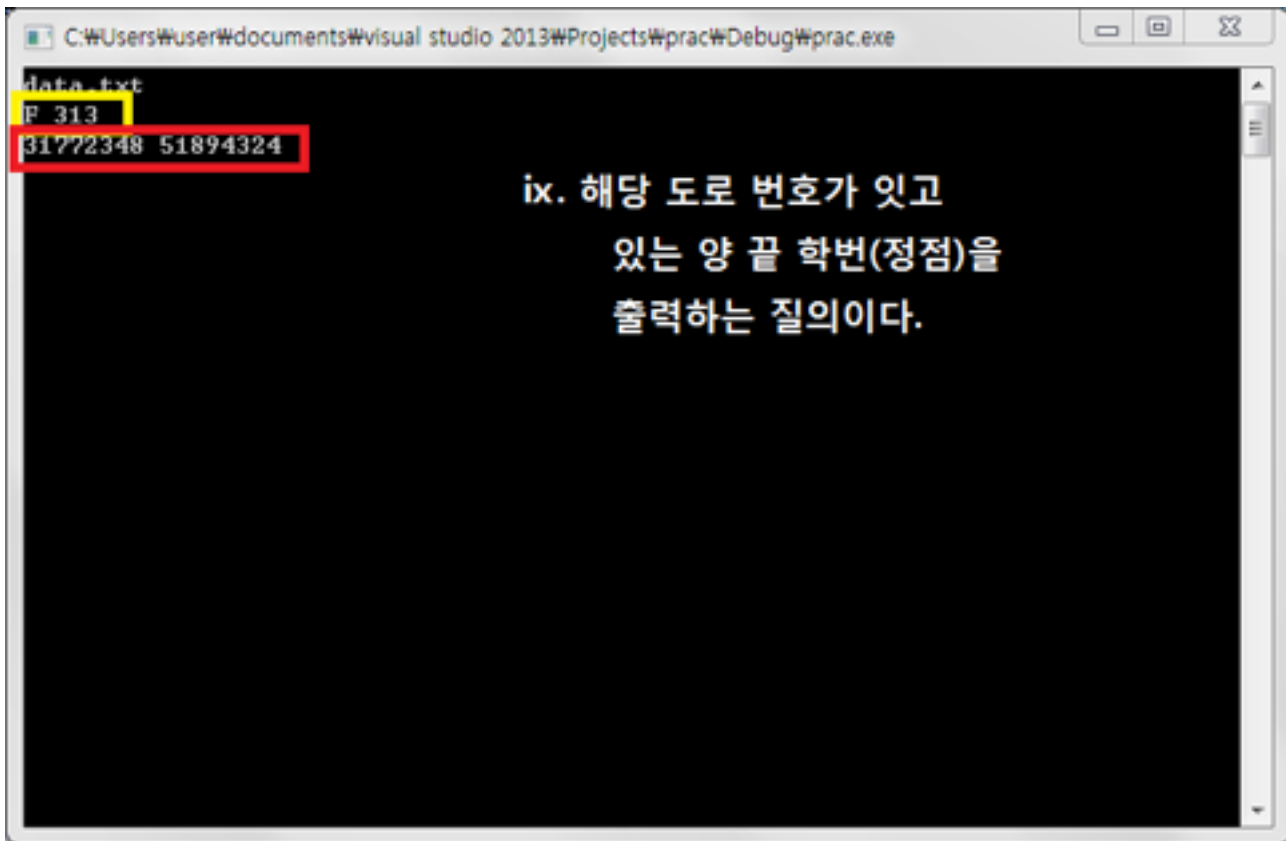
7) 연결된 도로 출력



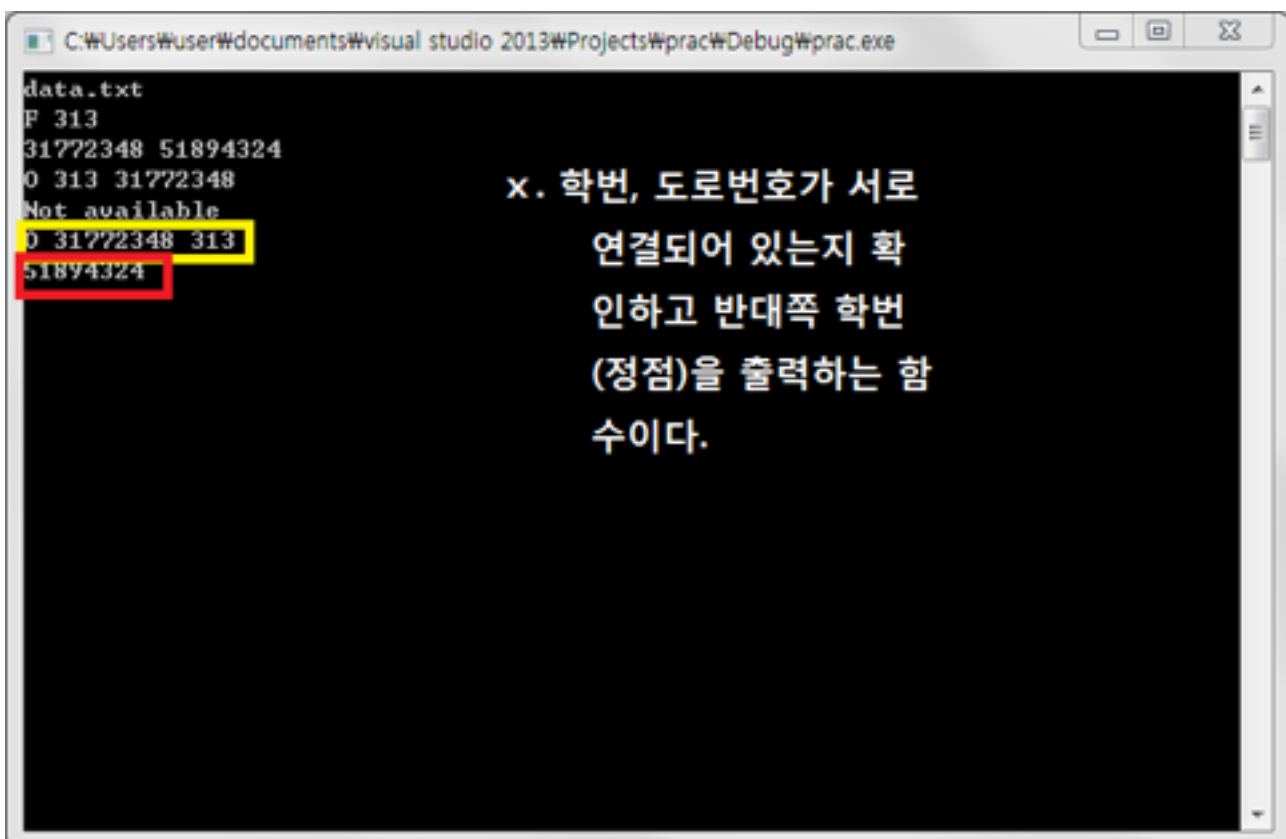
```
C:\Windows\system32\cmd.exe
data.txt
A 12141579 channee@sample.com
21
P 12141579
channee@sample.com 0
I 753 12141579 88521479
21
Z 753
20
K 12141579 88521479
Not found
L 88521479
11
```

viii. 해당 학번에 연결된  
도로 번호를 출력  
하는 질의이다.

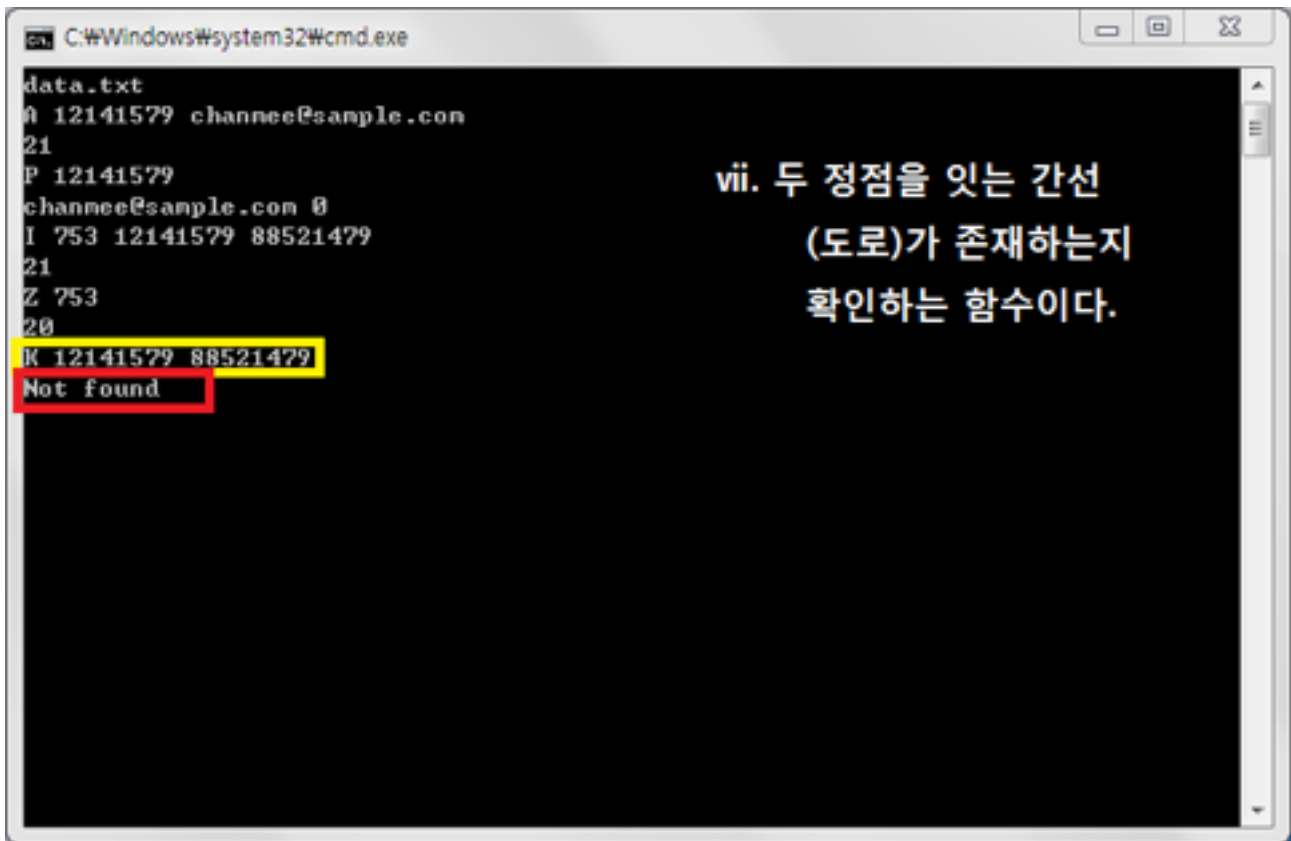
## 8) 도로 정보 출력 1



## 9) 도로 정보 출력 2



## 10) 도로 존재 여부 확인



```
C:\Windows\system32\cmd.exe
data.txt
Q 12141579 channee@sample.com
21
P 12141579
channee@sample.com 0
I 753 12141579 88521479
21
Z 753
20
K 12141579 88521479
Not found
```

vii. 두 정점을 잇는 간선  
(도로)가 존재하는지  
확인하는 함수이다.

## V. 평가 및 개선 방향

### - 본 결과의 장점 및 단점

#### 장점

- > Worst Case가 아닌 경우 빠른 시간에 기능이 수행 될 수 있다.
- > 각 정점의 차수를 입력해 주었기 때문에 빠른 시간 내에 할 수 있다.

#### 단점

- > cpp파일을 나누어 주지 않았기 때문에 가독성이 떨어진다.
- > 조건문이 너무 많이 쓰인다.
- > while 문을 통한 반복문이 많이 쓰인다.
- > 파일 입출력시 간선을 삽입하는 것과 질의를 통해 삽입하는 과정이 거의 동일함에도 불구하고 합치지 않았기에 코드가 길어졌다.
- > 정점 삭제를 할 수 없다.
- > 간선 삭제의 수행시간이 느리다.  $O(m)$ .

### -향후 개선 방향

- > 조건문의 사용을 줄여 코드를 간단하게 한다.
- > Header, cpp, main을 나누어 가독성을 좋게 만든다.
- > 같은 과정을 반복하는 것은 새로운 함수를 만들어 주어 짧은 코드를 만든다.
- > 정점 삭제를 해준다.