객체지향 프로그래밍 중간 프로젝트

- 개발목표: KHUIS 수강신청 시스템 구현

- 참고사항
  1. 헤더 수정 불가 (제공된 스켈레톤 코드에 포함된 라이브러리 외에 추가 불가)
  2. 제공된 소스파일의 함수명, 변수명 변경 불가
  3. 함수는 선언부와 정의부를 나누어 작성할 것
  4. 구현된 함수 및 완성도에 따라 점수 차등 지급
  5. 필요시 멘토링 제도를 적극 활용할 것
  6. 제출 파일: 소스코드(.cpp 파일) 및 시스템 설명도(PPT 5 페이지 내외, 다이어그램 활용)

- 요구사항

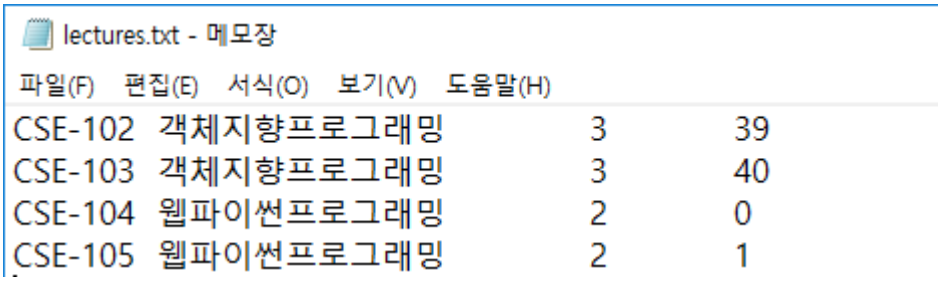
0) 데이터 불러오기

- 프로그램은 실행 시에 students.txt를 읽어 들여 vector의 형태로 저장한다. 하나의 레코드(아이디, 비밀번호, 이름, 수강가능학점, ~~수강신청과목수~~, 수강 신청한 과목)는 각각의 vector에 저장하도록 하며, 같은 인덱스에 위치하는 원소들끼리 레코드를 구성하도록 한다. (예: 아이디 vector내의 i번째 원소와 비밀번호 vector내의 i번째 원소가 짝이 되게 한다.)

- 파일 상세 예시 students.txt (각 필드는 '₩t'으로 구분되며, 공백을 포함하지 않는다.)

- 프로그램은 실행 시에 lectures.txt를 읽어 들여 vector의 형태로 저장한다. 하나의 레코드(과목코드, 과목명, 학점, 수강가능인원)는 각각의 vector에 저장하도록 하며, 같은 인덱스에 위치하는 원소들끼리 레코드를 구성하도록 한다. (예: i번째 과목코드에 상응하는 과목명은 i번째 과목명이다.)

- 파일 상세 예시 lectures.txt (각 필드는 '₩t'으로 구분되며, 공백을 포함하지 않는다.)

- 

-

- 제시된 예시 파일은 단순 예시일 뿐이지만, 후술될 기능에 대한 예시는 본 예시 파일을 기반으로 한다.

1) 로그인

1-1) 학생 로그인

- 사용자로부터 아이디를 입력 받는다.
- 사용자로부터 비밀번호를 입력 받는다.
- 입력된 정보를 바탕으로 아이디의 존재여부를 검사하고, 존재할 시에 비밀번호가 일치하는지를 검사한다.
- 성공적으로 로그인시 학생메뉴로 넘어간다.

1-2) 관리자 로그인

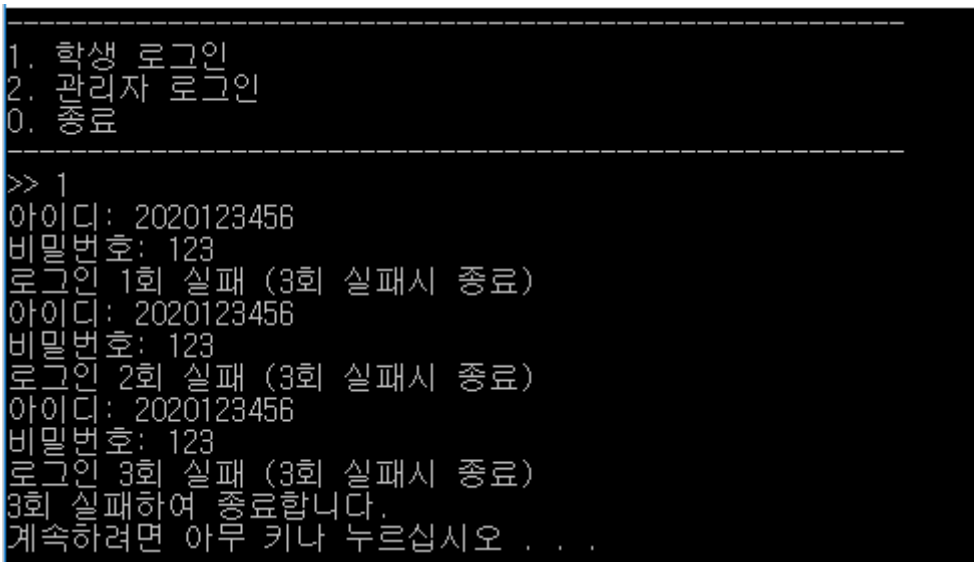
- 사용자로부터 아이디를 입력 받는다. (관리자 아이디는 파일로 관리하지 않으며,

"admin"으로 프로그램 내부에 존재한다.)

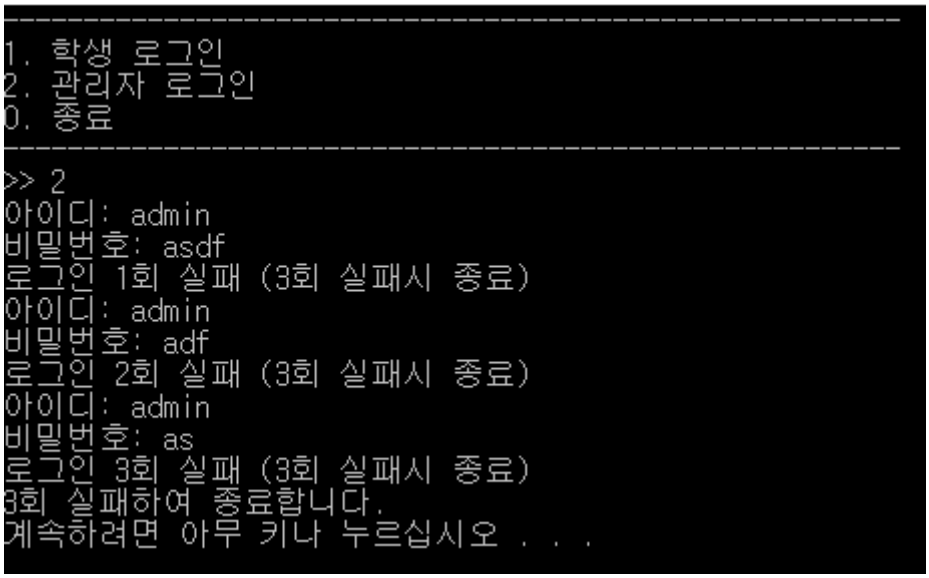
- 사용자로부터 비밀번호를 입력 받는다. (관리자 비밀번호는 파일로 관리하지 않으며, "admin"으로 프로그램 내부에 존재한다.)
- 입력된 정보가 일치할 시 관리자 메뉴로 넘어간다.

1-3) 로그인 실패

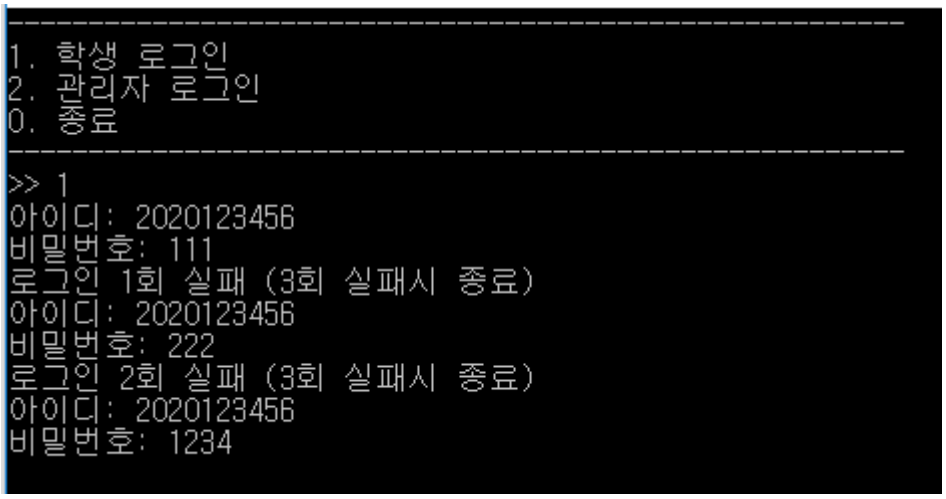
- 로그인 실패 시 "로그인 n회 실패 (3회 실패 시 종료)" 문구를 출력한다.
- 3회 실패 시 "로그인 3회 실패 (3회 실패 시 종료)" 문구를 출력한 후, "3회 실패하여 종료합니다." 문구를 출력 후 프로그램을 종료시킨다.

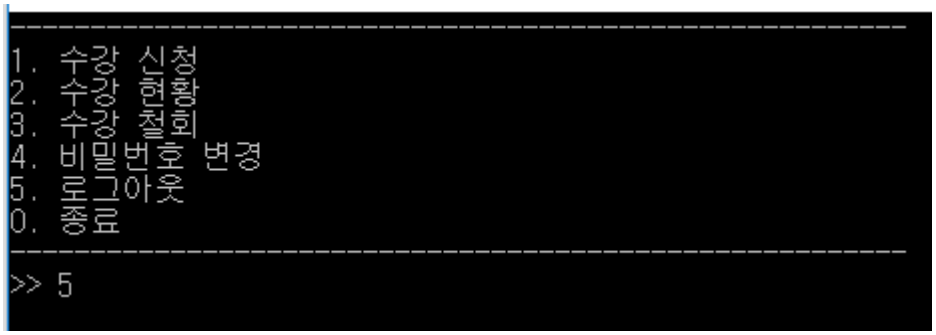


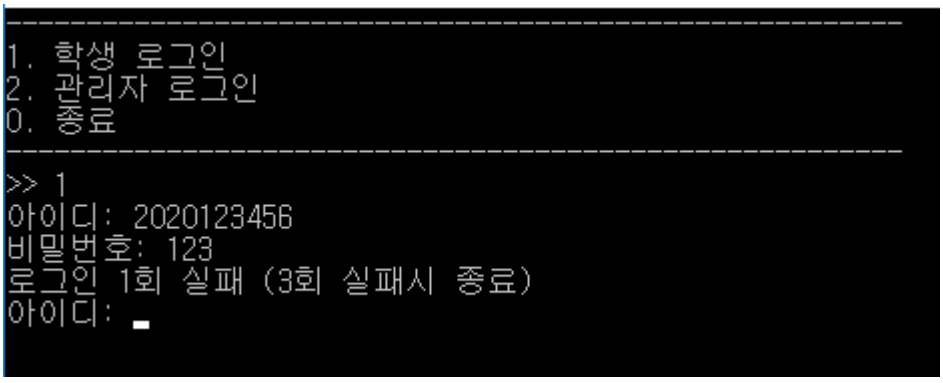
- 로그인 3회 미만 실패 후 로그인 성공 시에는 로그인 실패 횟수가 초기화 된다. 즉, 로그인 성공 후 로그아웃 후 다시 로그인 시도 시 실패횟수는 0부터 시작한다.



-







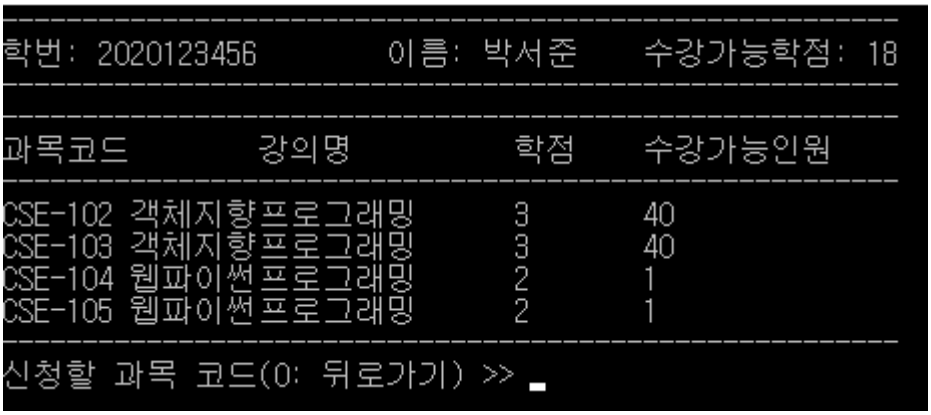
2) 학생 메뉴

2-1) 수강 신청

- 현재 로그인한 학생의 정보를 출력한다. "학번: {학번}₩t이름: {이름}₩t 수강가능학점: {수강가능학점}₩n"

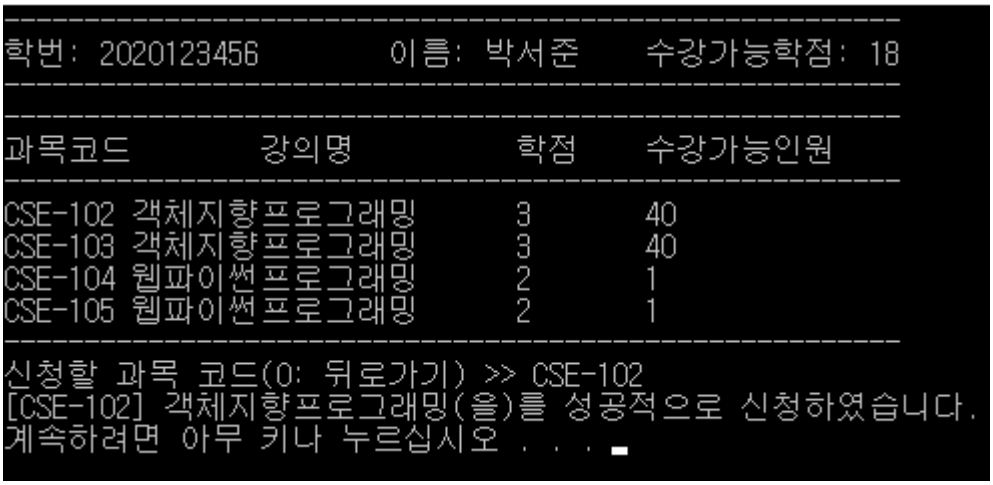
- 현재 개설 되어있는 모든 강의를 출력한다.

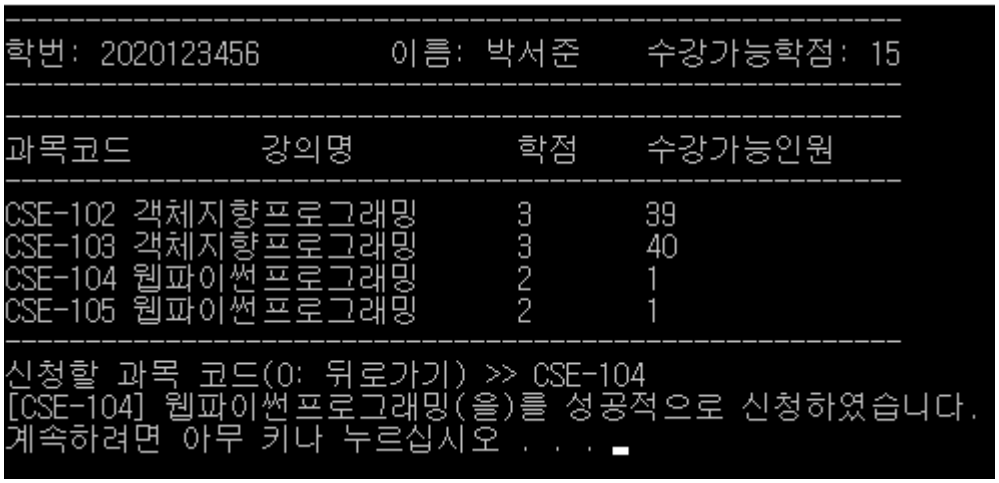
- 사용자로부터 수강 신청할 과목 코드를 입력 받는다.

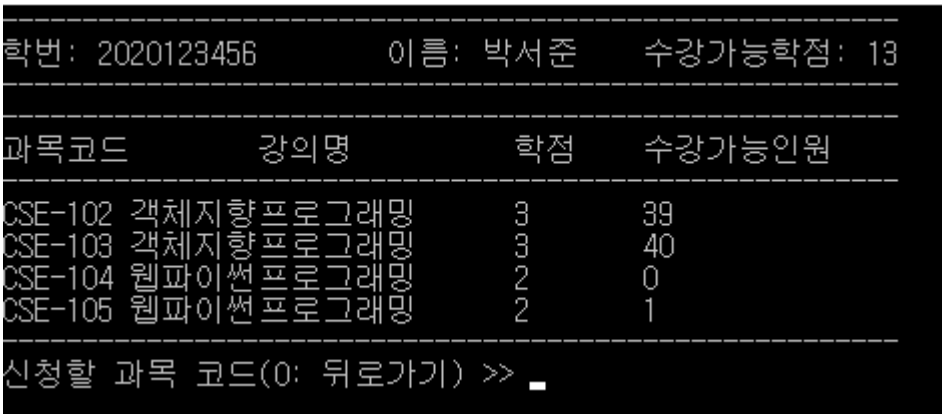


- 1) 동일 과목 코드를 가지는 과목을 이미 신청한 경우, 2) 동일 강의명을 가지는 과목을 이미 신청한 경우, 3) 수강가능학점이 신청하려는 과목보다 작은 경우, 4) 수강 가능인원이 0인 경우를 검사하고, 해당하지 않으면 성공적으로 신청되었다는 문구를 출력한다. "[{과목코드}] {강의명}(을)를 성공적으로 신청하였습니다."

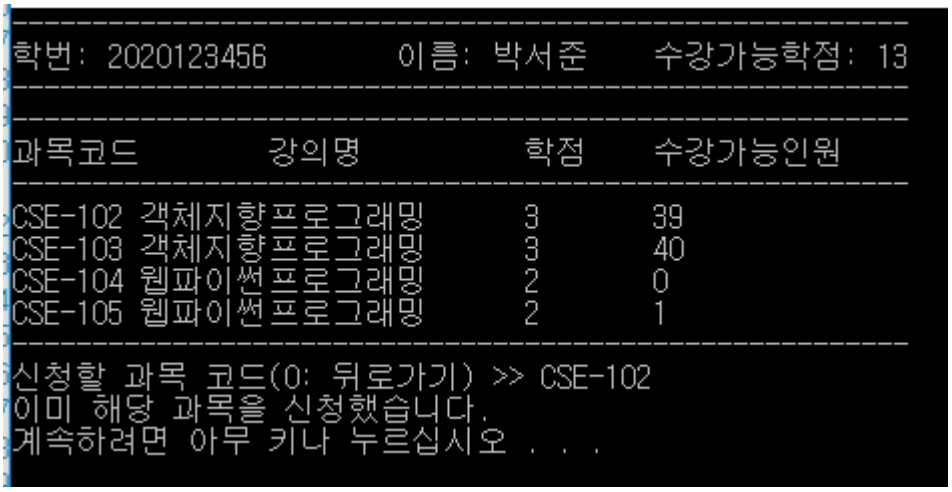
- 조건 검사 후 문구만 출력하는 것뿐만 아니라 실제로 해당 학생의 신청과목 vector에 신청한 과목 코드가 추가되어야 하고, 해당 학생의 수강 가능학점이 해당 과목의 학점만큼 차감되어야 하며, 해당 과목의 수강가능인원은 1만큼 줄어들어야 한다.



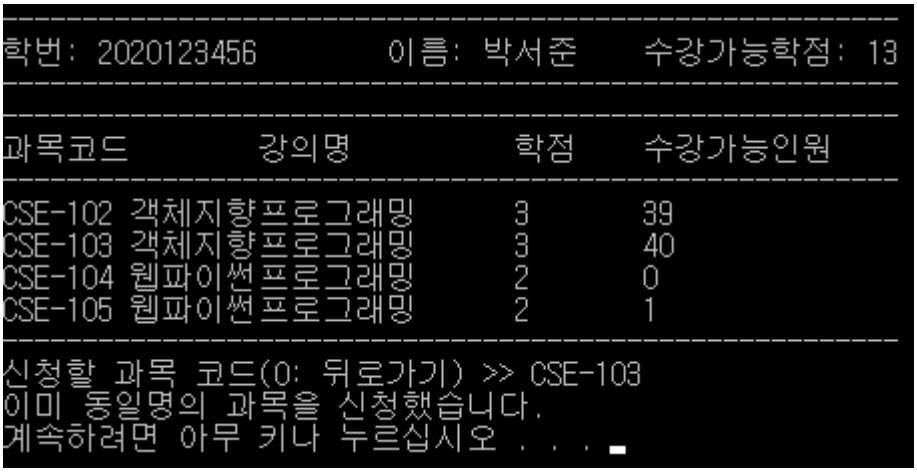




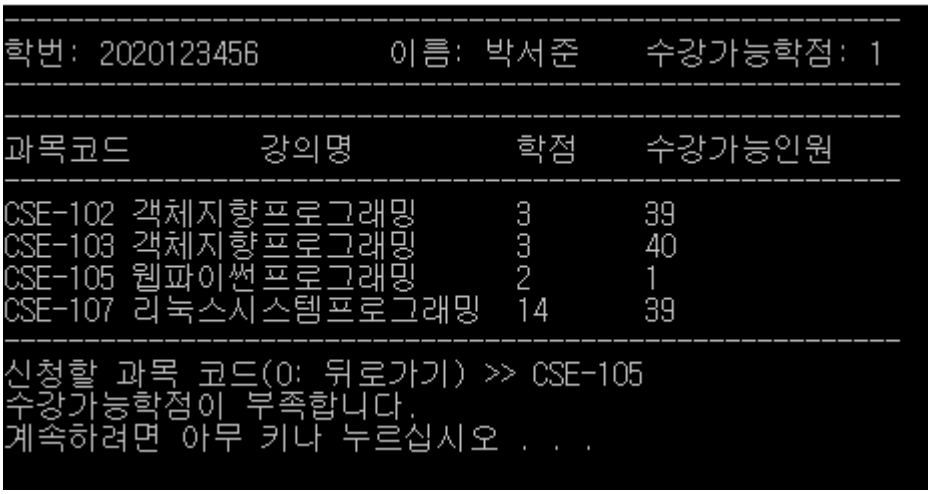
- 1) 동일 과목 코드를 가지는 과목을 이미 신청한 경우



- 2) 동일 강의명을 가지는 과목을 이미 신청한 경우

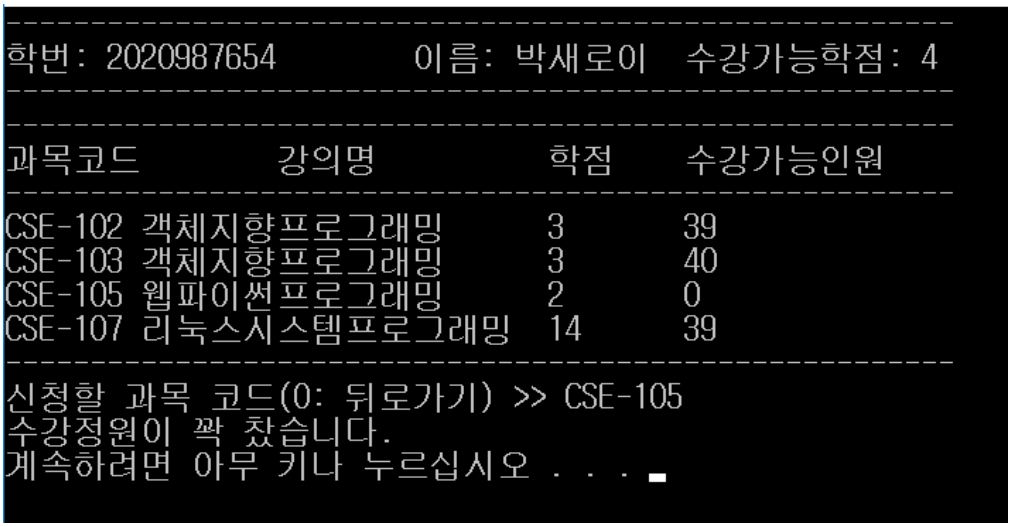


- 3) 수강가능학점이 신청하려는 과목보다 작은 경우



-

- 4) 수강 가능인원이 0인 경우



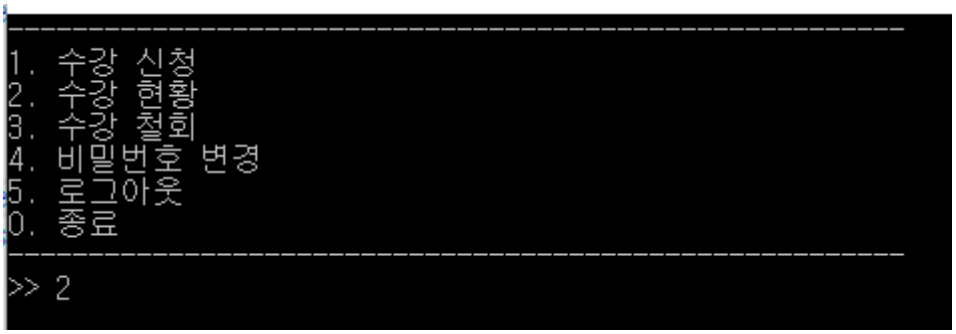
- 신청할 과목 코드에 해당하는 입력이 "0"이 아닌 경우를 제외하고는 계속 수강 신청 메뉴에 머무른다.

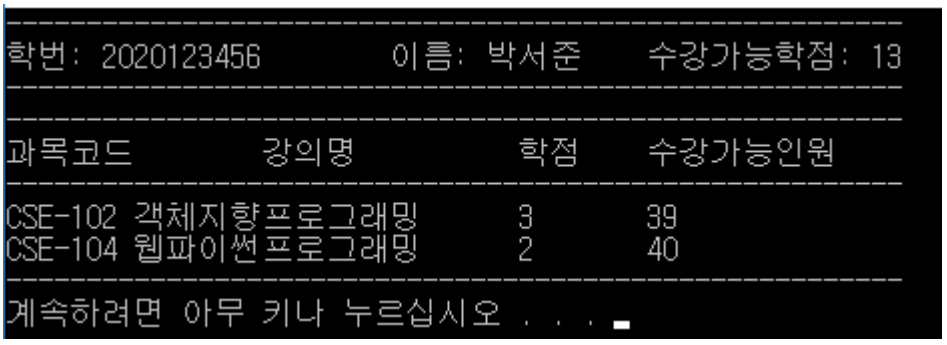
- 조건 검사 순서는 수강인원 충분여부 > 수강학점 충분여부 > 과목코드 중복여부 > 과목명 중복여부 순으로 진행한다. (예: 학점이 부족한 상태에서 신청할 과목 코드 입력 시 "수강 인원이 꽉 찼습니다." 출력

2-2) 수강 신청 현황

- 현재 학생의 정보를 출력한다.

- 수강 신청된 과목들을 출력한다. 과목코드, 강의명, 학점, 수강가능 인원





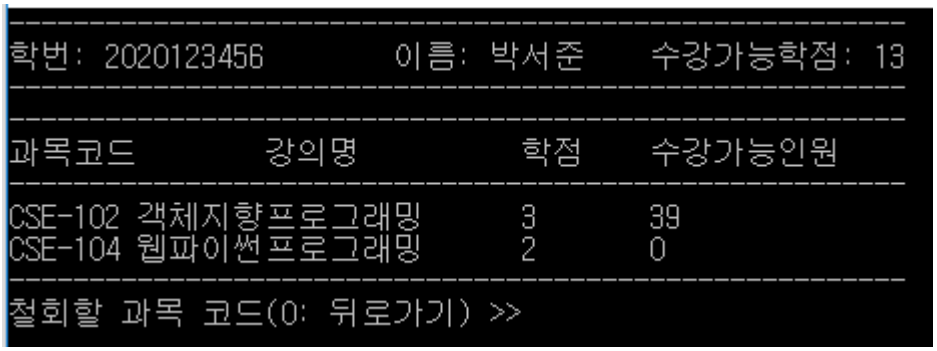
- 아무 키나 입력 시 다시 학생 메뉴로 돌아간다.

2-3) 수강철회

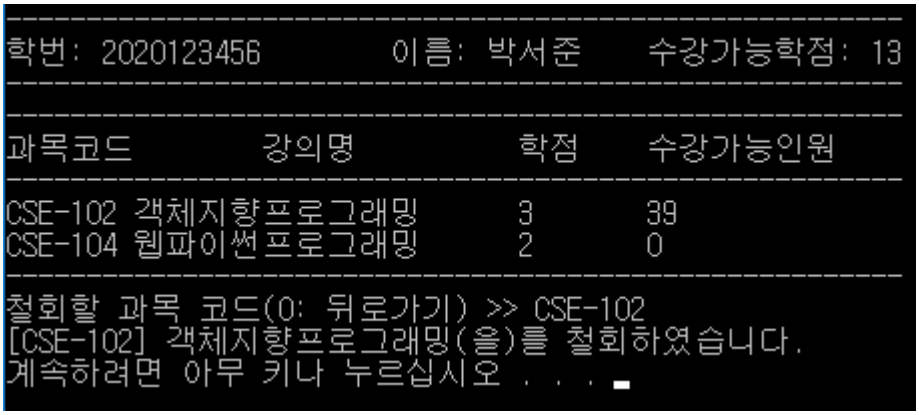
- 현재 학생의 정보를 출력한다

- 수강 신청된 과목들을 출력한다.

- 사용자로부터 철회할 과목 코드를 입력 받는다.

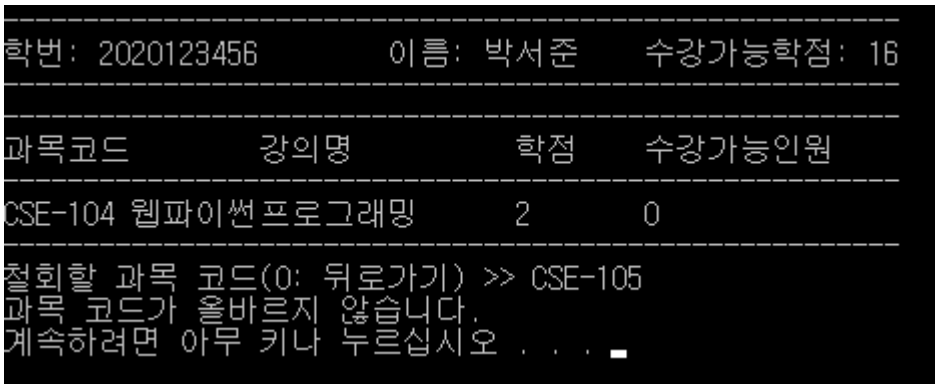


- 올바른 과목 코드 입력 시 "[{과목코드}] {과목명}(을)를 철회하였습니다."를 입력한다.

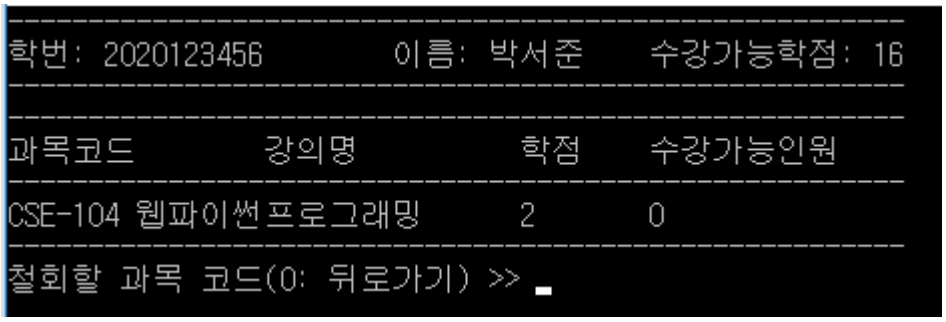


- 여기서 올바른 과목 코드란 현재 사용자가 취소할 수 있는 과목코드를 의미한다.

- 그 외에는 아래와 같이 처리한다.



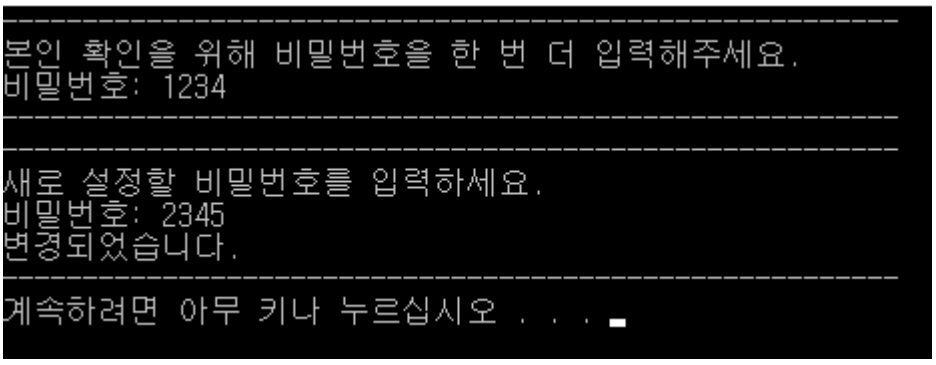
- 아무 키나 입력 시 해당 메뉴에 계속 머무른다



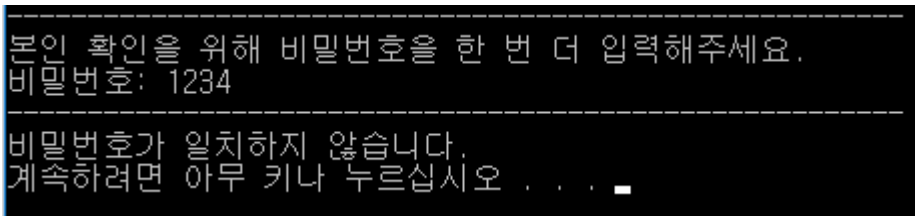
2-4) 비밀번호 변경

- "본인 확인을 위해 비밀번호를 한 번 더 입력해주세요." 라는 문구와 함께 현재 비밀번호를 입력 받는다.

- 일치하면, 새로 설정할 비밀번호를 입력한다.



- 일치하지 않으면, 다음과 같이 처리한다.



2-5) 로그아웃

- 단순하게 현재 접속한 사용자의 정보를 main에서 초기화 한다.

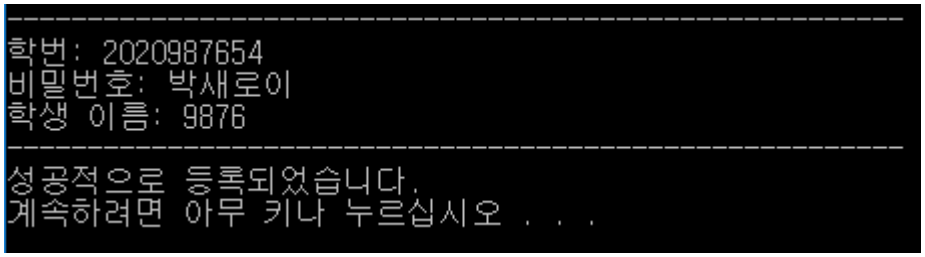
3) 관리자 기능

3-1) 학생 추가

- 수강신청 시스템은 임의의 사용자가 회원가입을 하는 것이 아니라, 관리자가 사용자를 추가하도록 설계 된다.

- 사용자로부터 학번(아이디), 비밀번호, 이름을 입력 받는다.

- 입력 받은 학번이 존재하지 않을 시에 성공적으로 등록된다.



- 입력 받은 학번이 존재할 시 이미 존재하는 학번임을 출력한다.

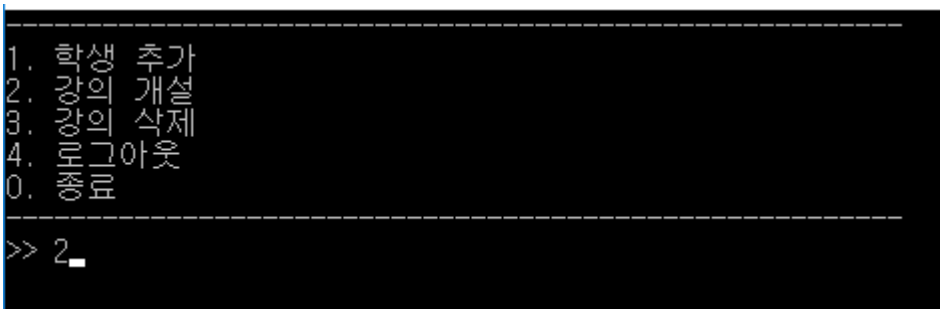
- 추가된 학생은 학번, 비밀번호, 학생 이름, 수강가능학점(고정 값: 18), 수강신청 과목목록(크기 0의 벡터)를 각각 알맞은 벡터에 추가하여야 한다.

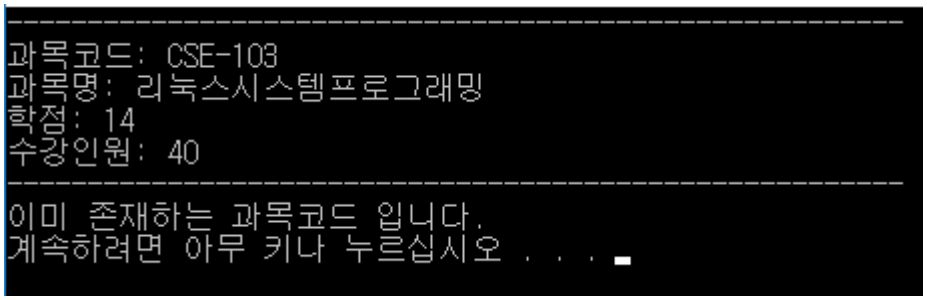
3-2) 강의 개설

- 강의 개설을 통해 개설된 강의는 학생 사용자에게 노출되며 수강신청 또한 가능하다.

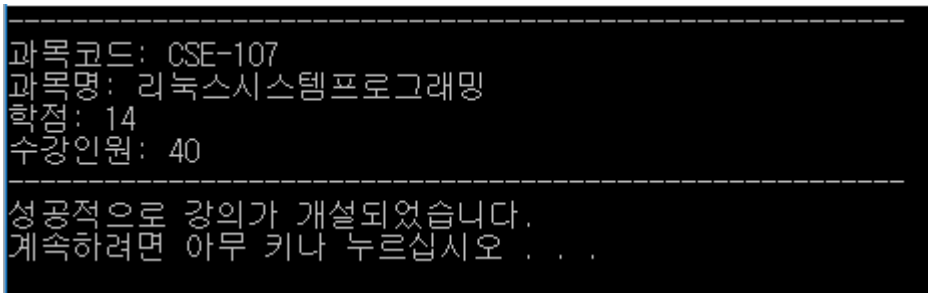
- 과목코드, 과목명, 학점, 수강인원을 차례대로 입력 받는다.

- 과목코드에 한해서 중복검사를 진행 후 과목코드가 이미 존재할 경우 아래와 같이 처리한다.





- 과목코드가 중복이 아니라면 성공적으로 강의를 개설했다는 문구를 출력한다.

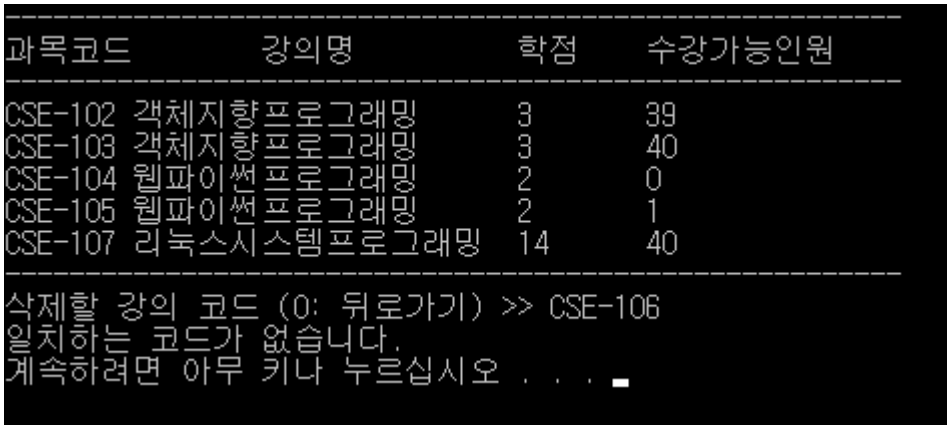


3-3) 강의 삭제

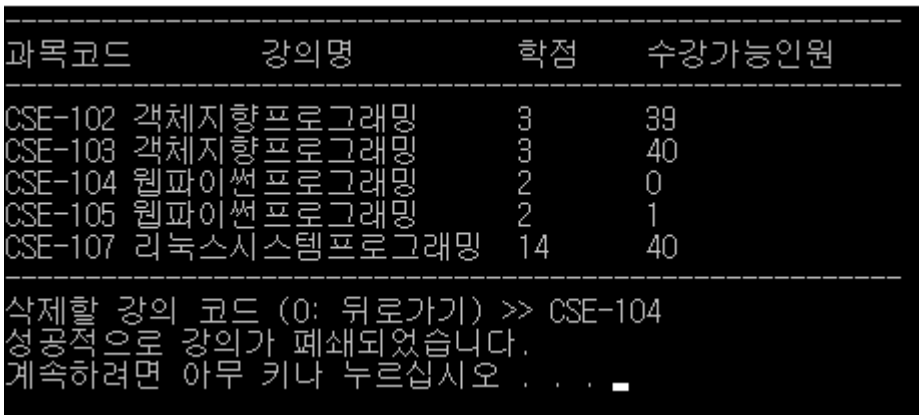
- 관리자는 강의를 삭제할(폐강시킬) 수 있다. 해당 과목을 수강하고 있는 학생이 있어도 삭제할 수 있다.

- 현재 개설되어 있는 강의의 목록을 출력하고, 사용자로부터 삭제할 강의 코드를 입력 받는다.

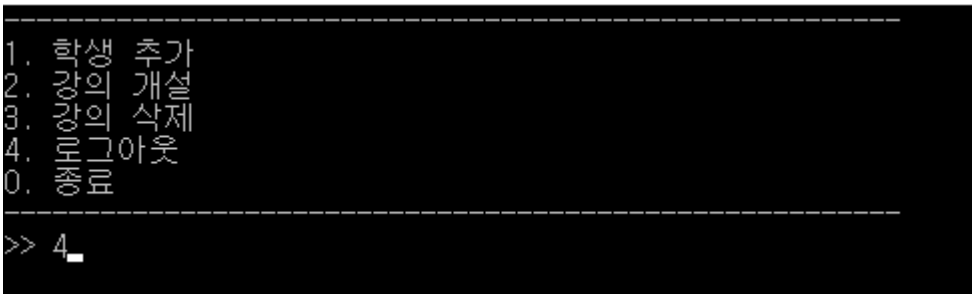
- 일치하는 과목 코드가 없는 경우 아래와 같이 처리한다.

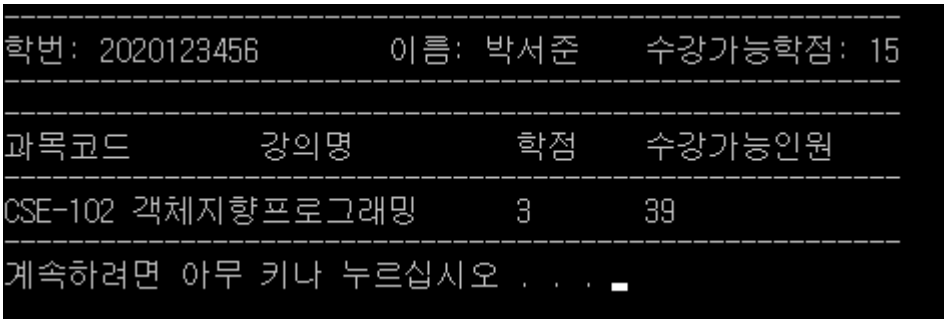


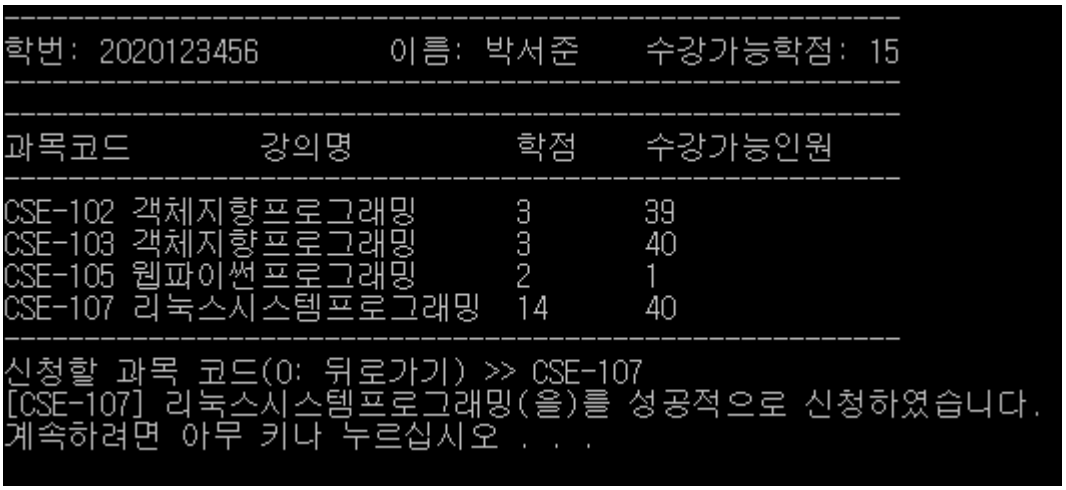
- 올바르게 입력한 경우 성공적으로 강의가 폐쇄되었음을 출력한다.



- 강의 폐쇄 시 해당 과목을 수강신청 과목 목록 내에 담고 있던 학생들은 해당 과목이 강제로 철회된다.



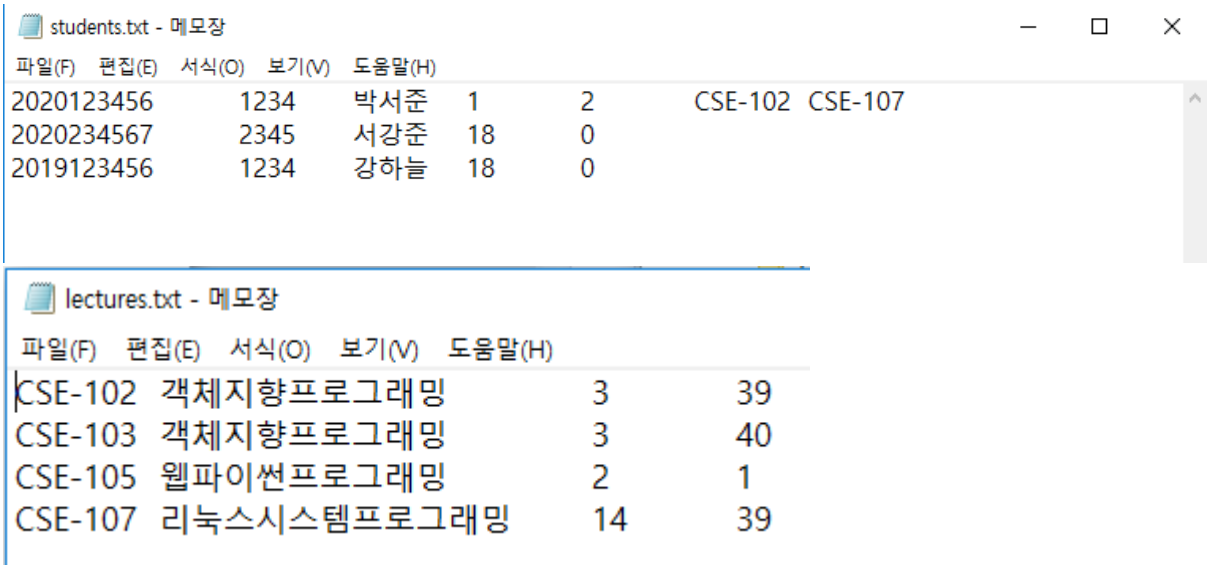




4) 데이터 내보내기

- 프로그램 이 작동하는 동안 변경된 모든 내용을 파일에 쓴다.

- 파일은 빈 상태로 열리며 처음부터 다시 쓴다.



- 기능별 관련 함수

사용되는 입력:

```
// Student Info
vector<string> studentIds; 모든 학생의 아이디(학번)를 저장하는 벡터
vector<string> passwords; 모든 학생의 비밀번호를 저장하는 벡터
vector<string> names; 모든 학생의 이름을 저장하는 벡터
vector<int> credits; 모든 학생의 수강가능학점을 저장하는 벡터
vector<vector<string>> appliedLectureCodes; 모든 학생의 수강신청현황 과목 벡터

// Lecture Info
vector<string> lectureCodes; 강의 코드 벡터
vector<string> lectureNames; 강의 명 벡터
vector<int> lectureCredits; 강의 학점 벡터
vector<int> limits; 강의 수강정원 벡터
```

추가) 기능 함수

```
int findStudentById(vector<string> studentIds, string id);
```
아이디로 학생의 인덱스를 찾아 반환
```
int findLectureByCode(vector<string> lectureCodes, string code);
```
과목코드로 과목 인덱스를 반환
```
void deleteElement(vector<string>& v, int index);
```

해당 인덱스에 해당하는 원소를 벡터에서 삭제

0) 데이터 불러오기

```
void readStudentFile(vector<string>& studentIds, vector<string>& passwords,
vector<string>& names, vector<int>& credits, vector<vector<string>>&
appliedLectureCodes);

void readLectureFile(vector<string>& lectureCodes, vector<string>& lectureNames,
vector<int>& lectureCredits, vector<int>& limits);
```

1) 로그인

```
string getInputId()
```
반환 값: 사용자 입력 아이디

```
string getInputPassword()
```
반환 값: 사용자 입력 비밀번호

```
int login(const vector<string>& studentIds, const vector<string>& passwords);
```
전체적인 로그인 과정을 관리하는 함수

반환 값: 학생 성공(학생 아이디 벡터 내에서의 인덱스), 관리자 성공(-100), 실패(-999)

1-1) 학생로그인

```
int studentLogin(const vector<string>& studentIds, const vector<string>&
passwords);
```
반환 값: 성공(학생 아이디 벡터 내에서의 인덱스), 실패(-1)

1-2) 관리자로그인

```
bool adminLogin()
```
반환 값: 성공(true), 실패(false)

2) 학생 메뉴

```
int runStudent(vector<string>& studentIds, vector<string>& passwords, vector<string>&
names, vector<int>& credits, vector<vector<string>>& appliedLectureCodes,
vector<string>& lectureCodes, vector<string>& lectureNames, vector<int>& lectureCredits,
vector<int>& limits, int user)
```
학생 메뉴를 총괄하는 함수

반환 값: 로그아웃(-1), 종료(1)

```
void printStudent(const vector<string>& studentIds, const vector<string>& names, const
vector<int>& credits, const int& user);
```

간단한 학생 정보를 출력한다. 학번, 이름, 수강가능학점. 학생메뉴에서 수강신청, 수강현황, 수강철회 메뉴에서 호출된다.

```
void printLectures(const vector<vector<string>>& appliedLectureCodes, const
vector<string>& lectureCodes, const vector<string>& lectureNames, const vector<int>&
lectureCredits, const vector<int>& limits, const int& user = -100);
```

개설된 강의들의 목록을 출력하거나(user 값이 -100 일 때) 현재 사용자가 신청한 수강과목 목록을 출력하는 함수(user 값이 -100 이 아닐 때)

2-1) 수강신청

```
void applyLecture(vector<string> studentIds, vector<string> names, vector<int>& credits,
vector<vector<string>>& appliedLectureCodes, vector<string> lectureCodes,
vector<string> lectureNames, vector<int> lectureCredits, vector<int>& limits, int user)
```

수강신청을 처리하는 함수이다.

2-2) 수강현황

```
printStudent(studentIds, names, credits, user);
printLectures(appliedLectureCodes, lectureCodes, lectureNames, lectureCredits, limits,
user);
```

개별 함수 대신 위에서 설명한 두 개의 함수 호출로 대체할 수 있다.

2-3) 수강철회

```
void disapplyLecture(const vector<string>& studentIds, const vector<string>& names,
vector<int>& credits, vector<vector<string>>& appliedLectureCodes, const
vector<string>& lectureCodes, const vector<string>& lectureNames, const vector<int>&
lectureCredits, vector<int>& limits, const int& user);
```

2-4) 비밀번호변경

```
void changePassword(vector<string>& passwords, const int& user);
```

2-5) 로그아웃

`int runStudent` 함수에서 `main` 으로 `-1` 을 반환한다.

2-6) 종료

`int runStudent` 함수에서 `main` 으로 `1` 을 반환한다.

3) 관리자 메뉴

```
int runAdmin(vector<string>& studentIds, vector<string>& passwords, vector<string>&
names, vector<int>& credits, vector<vector<string>>& appliedLectureCodes,
vector<string>& lectureCodes, vector<string>& lectureNames, vector<int>& lectureCredits,
vector<int>& limits)
```

관리자 메뉴를 총괄하는 함수이다.

반환 값: 로그아웃(-1), 종료(1)

3-1) 학생추가

```
void addStudent(vector<string>& studentIds, vector<string>& passwords, vector<string>&
names, vector<int>& credits, vector<vector<string>>& appliedLectureCodes)
```

3-2) 강의개설

```
void addLecture(vector<string>& lectureCodes, vector<string>& lectureNames,
vector<int>& lectureCredits, vector<int>& limits)
```

3-3) 강의삭제

```
void deleteLecture(vector<string>& lectureCodes, vector<string>& lectureNames,
vector<int>& lectureCredits, vector<int>& limits, vector<int>& credits,
vector<vector<string>>& appliedLectureCodes)
```

3-4) 로그아웃

int runAdmin 함수에서 main 으로 -1 을 반환한다.

3-5) 종료

int runAdmin 함수에서 main 으로 1 을 반환한다.

4) 데이터 내보내기

```
void writeStudentFile(const vector<string>& studentIds, const vector<string>& passwords,
const vector<string>& names, const vector<int>& credits, const vector<vector<string>>&
appliedLectureCodes);

void writeLectureFile(const vector<string>& lectureCodes, const vector<string>&
lectureNames, const vector<int>& lectureCredits, const vector<int>& limits);
```

-