

2023 전공기초프로젝트1 1차 기획서

# 주소록 관리 프로그램: KUdress book

3238 B01 팀

202211389 최준규

201911277 장세현

202211392 최 환

202211342 이율원

## 목차

<b>1 프로젝트 개요.....</b>	<b>4</b>
1.1 목적.....	4
1.2 대상.....	4
1.3 예상 기대효과.....	4
<b>2 용어.....</b>	<b>4</b>
<b>3 사용 흐름도.....</b>	<b>5</b>
<b>4 프로그램 내용.....</b>	<b>6</b>
4.1 주소록 요소.....	6
4.1.1 아이디.....	6
4.1.2 비밀번호.....	6
4.1.3 이름.....	7
4.1.4 전화번호.....	8
4.1.5 주소.....	9
4.1.6 생일.....	9
4.1.7 메모.....	11
4.2 프로그램 기능.....	12
4.2.1 로그인 및 회원가입.....	12
4.2.1.1 로그인.....	12
4.2.1.2 회원가입.....	13
4.2.1.3 로그아웃.....	14
4.2.2 주소록 열람.....	14
4.2.3 연락처 검색.....	17
4.2.4 연락처 추가.....	17
4.2.5 연락처 수정.....	18
4.2.6 연락처 삭제.....	19
4.2.7 내 정보 수정.....	20

---

4.2.8 도움말.....	21
4.2.9 종료.....	21
<b>5 데이터 파일.....</b>	<b>22</b>
5.1 문법 규칙.....	22
5.1.1 기본 규칙.....	22
5.1.2 회원 정보 데이터 파일 규칙 및 예시.....	23
5.1.3 개인별 데이터 파일 규칙 및 예시.....	23
5.2 의미 규칙.....	24
5.3 부가 확인 목록.....	24
5.4 무결성 확인 및 처리.....	24
<b>6 주 프롬프트.....</b>	<b>25</b>
6.1 도움말.....	27
6.2 종료.....	27
6.3 로그인 & 가입 명령어 군.....	27
6.3.1 로그인.....	27
6.3.2 회원가입.....	28
6.3.3 로그아웃.....	28
6.4 열람.....	28
6.5 추가.....	28
6.6 삭제.....	29
6.7 수정.....	29
6.8 검색.....	29
<b>7 예외 처리.....</b>	<b>29</b>
7.1 동명이인.....	29
7.2 중복 전화번호.....	29
7.3 이름이 정수(Integer).....	30
7.4 주소록 요소들에 탭 문자(\t) 존재.....	30
7.5 문법 오류.....	31

## 1 프로젝트 개요

### 1.1 목적

프로젝트의 목적은 사용자가 자신이 가진 주소록 정보를 보다 효율적이고 체계적으로 관리할 수 있도록 도와주는 CLI 기반 주소록 관리 프로그램을 개발하는 것이다. 이 프로그램은 아이디와 비밀번호를 통한 로그인 기능을 제공하며, 내 정보 보기 및 주소록 열람, 수정, 삭제, 검색 기능이 있다.

### 1.2 대상

이 프로그램은 기본적으로 건국대 학생을 대상으로 하나, 주소록을 관리하고 싶어하는 사용자라면 누구나 사용이 가능하다.

### 1.3 예상 기대효과

이 프로그램을 통해 사용자가 CLI 기반 환경을 통해 빠르고 정확하게 주소록 정보를 관리할 수 있을 것으로 기대된다.

## 2 용어

- **프로그램:** 이 문서를 통해 기획·명세하고있는 대상 프로그램인 “KUdress Book” 프로그램
- **사용자:** 프로그램을 사용하는 주체
- **연락처:** 이름, 전화번호, 주소, 생일, 메모를 가진 객체
- **주소록:** 연락처들을 모아둔 목록
- **내 정보:** 사용자의 이름, 전화번호, 주소, 생년월일을 가진 객체 (연락처와 달리 메모가 없음)
- **연락처 열람 상태:** 하나의 연락처를 열람해 주소록 수정, 삭제를 할 수 있는 상태
- **로그인:** 등록되어있는 사용자의 아이디와 비밀번호를 입력받고 해당 사용자의 주소록을 불러옴
- **회원가입:** 사용자를 새로 등록해 새로운 주소록을 생성
- **도움말:** 사용자가 입력할 수 있는 명령어들과 그에 대한 설명
- **<value>:** 사용자의 입력값으로서, 문자형 데이터 또는 정수형 데이터이다.
- **페이지:** 출력되는 연락처의 목록이 10개를 초과할 때, 10개씩 나누어 구분하고 출력한다. 이때 구분된 연락처들의 단위를 페이지라고 한다.
- **로그인/회원가입 창:** 프로그램에서 로그인이나 회원가입을 요구하는 상태
- **숫자:**  $0^{U+0030} - 9^{U+0039}$  만을 의미하며, 아랍 숫자, 데바나가리 숫자, 로마 숫자 등 이외의 숫자는 포함되지 않음
- **공백:** U+0020을 의미하며, 아래의 문법 예시에서는 시인성을 위해 ‘ ’로 표기한다

- Tab문자: U+0009를 의미하며, 아래 문법 예시에서는 시인성을 위해 ‘→’로 표기한다

### 3 사용 흐름도

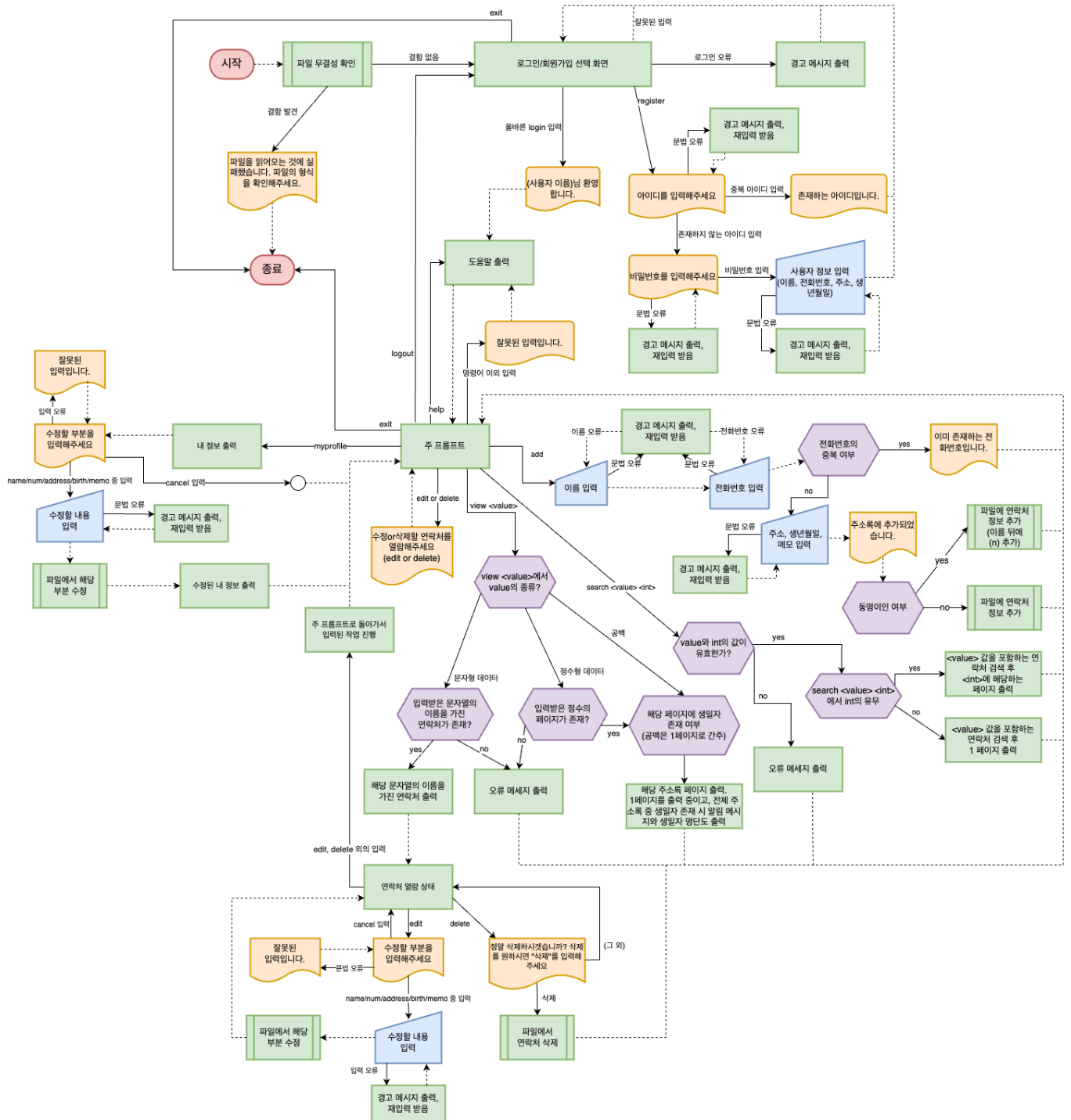


그림 1. 사용 흐름도

## 4 프로그램 내용

### 4.1 주소록 요소

주소록 요소에는 크게 아이디, 비밀번호, 이름, 전화번호, 주소, 생일, 메모가 있다.

각 요소마다 동치 비교 규칙이나, 검색 시의 검색어 합치<sup>Matching</sup> 규칙도 서로 다르므로, 아래의 각 요소별 소절에서 해당 요소의 동치 비교 규칙과 검색어 합치 규칙을 각각 명시한다. 단, 정렬시의 순서는 Lexicographic Order를 따른다.

#### 4.1.1 아이디

아이디는 사용자가 로그인에 사용할 아이디다. 사용자를 식별하기 위한 Unique한 값이다.

**문법 형식:** 문법적으로 올바른 아이디는 아래 두 조건을 모두 만족시키는 문자열이다:

- 길이가 5 이상
- 숫자('0' - '9') 또는 알파벳 대소문자('a' - 'z', 'A' - 'Z')로만 구성되어야 함

**의미 규칙:** 사용자를 식별하기 위해 Unique해야 한다.

**동치 비교:** 두 아이디 간의 동치를 비교할 때에는, 두 이름 간의 동치 비교를 할 때에는, 두 문자열 전체에 포함된 모든 문자(공백 포함)의 순서와 내용이 서로 완전히 일치해야만 같은 아이디으로 간주한다.

**검색 규칙:** 아이디는 로그인 및 회원가입에서만 사용되고, 검색에서 사용되지 않는다.

#### 4.1.2 비밀번호

비밀번호는 사용자가 로그인에 사용할 비밀번호다. 로그인하려는 사용자가 아이디의 소유주와 동일한지 확인하는 용도로 사용된다.

**문법 형식:** 문법적으로 올바른 비밀번호는 아래 모두 만족시키는 문자열이다:

- 길이가 8 이상
- Whitespace를 포함할 수 없음
- 알파벳 대소문자('a' - 'z', 'A' - 'Z'), 숫자('0' - '9') 또는 특수문자로만 구성되어야 함

**의미 규칙:** 비밀번호에는 아무런 추가적인 의미 규칙이 없다.

**동치 비교:** 두 비밀번호 간의 동치를 비교할 때에는, 두 이름 간의 동치 비교를 할 때에는, 두 문자열 전체에 포함된 모든 문자(공백 포함)의 순서와 내용이 서로 완전히 일치해야만 같은 비밀번호로 간주한다.

**검색 규칙:** 비밀번호는 로그인 및 회원가입에서만 사용되고, 검색에서 사용되지 않는다.

#### 4.1.3 이름

이름은 말 그대로 이름이다. 사람의 이름이나, 기업의 이름 등이 될 수 있으며, 심지어 개개인을 식별하기 위한 아무 정보가 포함되어도 된다. 또한 동명이인의 경우가 있을 수 있으므로, 이름의 중복은 허용된다. 필수 입력 사항이다.

**문법 형식:** 문법적으로 올바른 이름은 아래 4개의 조건을 모두 만족시키는 문자열이다:

- 길이가 1 이상
- 첫 문자와 마지막 문자는 실상 문자
- 숫자만으로 구성된 문자열은 이름이 될 수 없음
- 탭 문자나 개행 문자를 포함하지 않음

예를 들어, ‘최환’과 ‘최준규(건컴22)’, ‘비빔밥 미래인재양성부 장관 이율원’, 심지어 ‘ㅇ’, ‘!@^#@’ 모두 문법적으로 올바른 이름이다.

**의미 규칙:** 이름에는 아무런 추가적인 의미 규칙이 없다.

**동치 비교:** 두 이름 간의 동치 비교를 할 때에는, 두 문자열 전체에 포함된 모든 문자(공백 포함)의 순서와 내용이 서로 완전히 일치해야만 같은 이름으로 간주한다.

예를 들어, ‘최환’과 ‘최 환’은 다른 이름이고, ‘Sean Storey’와 ‘sean storey’도 다른 이름이다.

**검색 규칙:** 검색 명령의 인자로서 이름을 활용하여 주소록에 저장된 연락처를 찾을 때에는:

- 저장된 이름 속의 공백류들을 전부 무시하고(공백류 없이 모두 붙어있는 문자열로 간주하고)
- 대소문자를 구분하지 않으면서
- 검색어가 이름의 부분문자열이거나 검색어가 초성만으로 이루어졌을 경우, 이름에 해당 초성이 포함되어 있을 경우 합치된 것으로 간주한다.

예를 들어 검색어가 ‘최준규’일 때, 저장된 연락처의 이름들 중 ‘최준규’뿐만 아니라 ‘최 준규’, ‘최준 규’도 모두 합치된다. 또한 ‘비빔밥’이 검색어일 때, 저장된 연락처의 이름들 중 ‘비빔밥 미래인재양성부 장관 이율원’, ‘제육비빔밥

전문 위락식당', '유비빔 밥집' 모두 합치된다. 또한, 검색어가 '췌ᄃᆞᆫ'일 경우 저장된 연락처의 이름 중 '최환', '췌환', '최환' 모두 합치된다. 또한 검색어가 'Sean Storey'일 경우, 저장된 연락처의 이름 중 'Sean storey', 'se anst orey' 모두 합치된다.

참고로, 대소문자를 구분하지 않을 경우는, 영문자 A-Z, a-z 만 해당된다. 그리스 문자, 키릴 문자 등 영어 알파벳 대소문자 52자를 제외한 모든 문자에 대해서는 대소문자를 철저히 구분한다.

#### 4.1.4 전화번호

이 프로그램은 '전화번호' 요소로 휴대전화 번호와 유선 전화 번호를 모두 표현할 수 있다. 필수 입력 사항이다.

**문법 형식:** 문법적으로 올바른 전화번호는 아래 6개의 조건을 모두 만족시키는 문자열이다.

- 1개 이상의 숫자들과 0개 이상의 '⁠<sup>U+002D</sup>' 문자들만으로 구성되어야 함(횡공백류열은 포함하지 않음)
- 첫 문자와 마지막 문자는 숫자여야 함
- '⁠<sup>U+002D</sup>'는 연속되어 나올 수 없음
- 처음 세 문자가 '010'이면, 이후 숫자는('⁠<sup>U+002D</sup>'가 몇 개든 상관 없이) 정확히 8개여야만 함
- 처음 두 문자가 '01'로 시작하고, 세번째 문자가 '0'이 아니면, 이후 숫자는('⁠<sup>U+002D</sup>'가 몇 개든 상관 없이) 7개 혹은 8개가 있어야 함

예를 들어, '010-8496-3744'나, '01087399719', '021234567', '112', '011-23456789', '1'은 모두 올바른 전화번호지만, '123-', '011--123-4567', '1 2 34 9'는 올바른 전화번호가 아니다.

**의미 규칙:** 전화번호에는 아무런 추가적인 의미 규칙이 없다.

**동치 비교:** 두 전화번호 간 동치 비교를 할 때에는, 두 문자열 내부의 '⁠<sup>U+002D</sup>'를 모두 무시한 채로, 그 문자열 전체가 서로 완전히 일치해야만 같은 전화번호로 간주한다.

예를 들어, '010-8496-3744'와 '01084963744'는 서로 같은 전화번호지만, '010-8496-3744'와 '8496-3744'는 서로 다른 전화번호이다.

**검색 규칙:** 검색 명령의 인자로서 전화번호를 활용하여 주소록에 저장된 연락처를 찾을 때에는, 검색어와 저장된 전화번호 내부의 '⁠<sup>U+002D</sup>' 문자를 모두 무시한 채로, 검색어가 전화번호의 부분문자열이면 합치된 것으로 간주한다.



예를 들어, 검색어가 '112'일 경우, 저장된 전화번호들 중 '112'와 '010-1123-4567'뿐만 아니라 '011-234-5678', '031-123-4567'까지도 모두 합치된다.

#### 4.1.5 주소

주소는 개인이나 업체의 집 주소, 또는 사무실 주소 등을 담기 위한 필드이다. 필수 입력 사항은 아니며, 사용자의 선택에 따라 주소를 입력하거나 입력하지 않을 수 있다.

**문법 형식:** 문법적으로 올바른 주소는 아래 세 조건에 모두 부합하는 문자열이다:

- 공백(U+0020)을 제외한 모든 윗공백류열(탭, 개행 등)을 사용할 수 없음
- 1개 이상의 문자, 숫자, 공백 문자들로만 구성되어야 함
- 공백 문자는 연속되어 나올 수 없음

위 규칙은 주소의 형식을 제한하지만, 현실적인 주소의 형식에는 제약이 있다. 따라서, 위 규칙에 따라 사용자의 입력을 처리한 후에도, 주소가 현실적인 형식에 부합하는지를 검증해야 하지만, 별도의 검증 없이 사용하도록 한다.

**의미 규칙:** 주소에는 아무런 추가적인 의미 규칙이 없다.

**동치 비교:** 두 주소 간 동치 비교를 할 때에는, 두 문자열 내부의 공백(U+0020)을 모두 무시한 채로, 그 문자열 전체가 서로 완전히 일치해야만 같은 주소로 간주한다.

예를 들어, '광진구 능동로120'과 '광진구 능동로 120'은 같은 주소이지만, '서울특별시 광진구 능동로 120'과 '광진구 능동로 120'은 다른 주소이다.

**검색 규칙:** 검색 명령의 인자로서 주소를 활용하여 주소록에 저장된 연락처를 찾을 때에는, 검색어와 저장된 주소 내부의 공백(U+0020)문자를 모두 무시한 채로, 검색어가 주소의 부분문자열이면 합치된 것으로 간주한다.

예를 들어, 검색어가 '광진'일 경우, 저장된 주소들 중 '광진구 능동로 120', '서울시 광진구 아차산로 1', '대구 수성구 광진슈퍼' 모두 합치된다.

#### 4.1.6 생일

생일은 개인의 생일 정보를 담기 위한 필드로, 필수 입력 사항은 아니며, 사용자의 선택에 따라 생일을 입력하거나 입력하지 않을 수 있다.

**문법 형식:** 문법적으로 올바른 생일은 아래 25개의 조건에 모두 부합하는 문자열이다:

- 숫자, <sup>U+002F</sup>, <sup>U+002D</sup>, <sup>U+002E</sup>, 으로만 구성되어야 함
- 연도, 월, 일 순으로 구성되어야 함
- 연도는 숫자 4개, 월, 일은 숫자 한 개 또는 두 개로 구성되어야 함
- 연도, 월, 일은 <sup>U+002F</sup>, <sup>U+002D</sup>, <sup>U+002E</sup>, 로 구분되거나 모두 붙여써야 함
  - 모두 붙여쓸 경우 월, 일은 숫자 두 개로만 구성되어야 함
- 연도의 첫 자리는 숫자 ‘1’ 또는 ‘2’만 허용되고, 이외의 자리는 ‘0’ - ‘9’가 모두 허용됨
- 월이 숫자 두 개로 구성되어 있을 때:
  - 첫 자리가 ‘0’일 경우, 둘째 자리는 ‘0’-‘9’로만 이루어져야 함
  - 첫 자리가 ‘1’일 경우, 둘째 자리는 ‘1’ 또는 ‘2’로만 이루어져야 함
- 월이 숫자 하나로 구성되어 있을 때:
  - ‘1’ - ‘9’로만 이루어져야함
- 일이 숫자 두 개로 구성되어 있을 때:
  - 첫 자리가 ‘0’일 경우, 둘째 자리는 ‘1’ - ‘9’로만 이루어져야 함
  - 첫 자리가 ‘1’일 경우, 둘째 자리는 ‘0’ - ‘9’로만 이루어져야 함
  - 첫 자리가 ‘2’이면서,
    - 2월이면서,
      - 연도가 윤년일 경우, 둘째 자리는 ‘0’ - ‘9’로만 이루어져야 함
      - 연도가 평년일 경우, 둘째 자리는 ‘0’ - ‘8’로만 이루어져야 함
    - 2월이 아닐 경우, 둘째 자리는 ‘0’ - ‘9’로만 이루어져야함
  - 첫 자리가 ‘3’이면서,
    - 월이 ‘1’, ‘01’, ‘3’, ‘03’, ‘5’, ‘05’, ‘7’, ‘07’, ‘8’, ‘08’, ‘10’, ‘12’일 경우, 둘째 자리는 ‘0’ 또는 ‘1’로만 이루어져야 함
    - 월이 ‘4’, ‘04’, ‘6’, ‘06’, ‘9’, ‘09’, ‘11’일 경우, 둘째 자리는 ‘0’으로만 이루어져야 함
  - 단, 월이 ‘2’ 또는 ‘02’일 경우는 첫 자리를 ‘3’으로 가질 수 없음
- 일이 숫자 하나로 구성되어 있을 때:
  - ‘1’ - ‘9’로만 이루어져야함

예를 들어, '2003/6/12', '2003.06.12', '2017-5-11', '20170511', '10000101'은 모두 올바른 생일이지만, '030612', '2017511', '2023/02/29', '20111355', '00000000'은 올바른 생일이 아닙니다.

**의미 규칙:** 생일에는 아무런 추가적인 의미 규칙도 없다. (의도한 모든 규칙을 문법으로 전부 표현했다.)

**동치 비교:** 두 생일 간의 동치 비교를 할 때에는, 두 문자열에서 연도, 월, 일을 추출하여 Leading Zero를 무시한 후 각각 비교하되, 서로 완전히 일치해야만 같은 생일로 간주한다.

예를 들어, '2003/06/12', '20030612', '2003.6.12'는 모두 같은 생일이지만, '2003/06/12'와 '2004/06/12'는 다른 생일이다.

**검색 규칙:** 검색 명령의 인자로서 생일을 활용하여 주소록에 저장된 연락처를 찾을 때에는, 검색어와 저장된 생일 문자열의 Seperator를 제거하고, 월, 일에 Leading Zero를 붙여 각각 두 글자로 만들어(yyyyMMdd)포맷으로 만들어 검색어가 이와 앞에서부터 순서대로 비교하여 같으면 합치된 것으로 간주한다.

예를 들어, 검색어가 '2003/6/12'이면 '20030612', '2003.06.12' 모두 합치되고, 검색어가 '2003.06'이면 '2003/6/12', '2003/06/27' 모두 합치되고, 검색어가 '2002.1'이면 '2002/1/23', '2002-01-25' 모두 합치되지만, '2002.12.25'는 합치되지 않는다.

#### 4.1.7 메모

메모는 개인의 기타 정보를 담기 위한 공간으로, 필수 입력 사항은 아니며, 사용자의 선택에 따라 메모를 입력하거나 입력하지 않을 수 있다.

**문법 형식:** 문법적으로 올바른 메모는 아래 n개의 조건에 모두 부합하는 문자열이다:

- 탭 문자, 공백, 문자, 숫자로만 구성되어야 함
- 1개 이상의 character로 구성되어야 함
- 첫 문자와 마지막 문자는 실상 문자여야 함

예를 들어 '건کم □ 멋쟁이', '건کم → 2학년 → 과대' 등은 모두 올바른 메모이지만, '비빔밥 (개행) 미래인재양성부장관', '□ 건대 □ 스용공'은 문법적으로 올바른 메모가 아니다.

**의미 규칙:** 메모에는 아무런 추가적인 의미 규칙도 없다.

**동치 비교:** 두 메모 간 동치 비교를 할 경우, 두 문자열 전체가 서로 완전히 일치해야만 같은 메모로 간주된다.

**검색 규칙:** 검색 명령의 인자로서 메모를 활용하여 주소록에 저장된 연락처를 찾을 때에는, 검색어와 저장된 메모 문자열의 모든 황공백류열을 무시한 채로, 검색어가 메모의 부분문자열이면 합치된 것으로 간주한다.

예를 들어, ‘비빔밥’이 검색어일 경우, 저장된 메모 중 ‘비빔밥 회장’과 ‘건대 비빔밥 맛집’뿐만 아니라, ‘비빔밥부 회장’까지도 모두 합치된다.

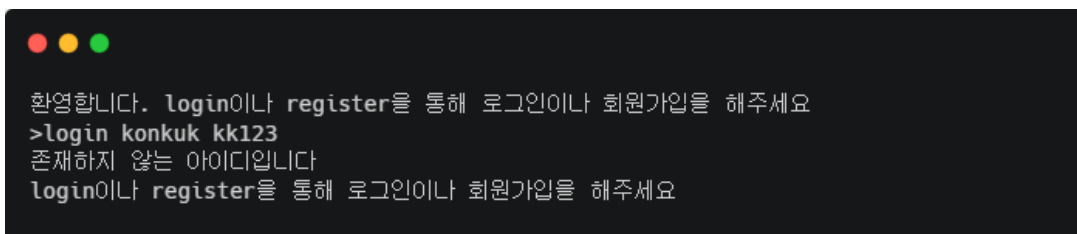
## 4.2 프로그램 기능

### 4.2.1 로그인 및 회원가입

- 프로그램 시작 시 “login이나 register을 통해 로그인이나 회원가입을 해주세요” 가 출력된다 ... (1)
- “login”, “register”, “exit” 이외의 입력에는 전부 (1)로 돌아간다.

#### 4.2.1.1 로그인

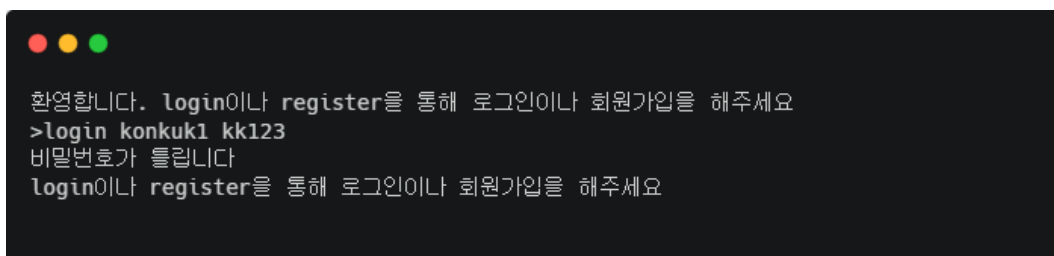
- “login (아이디) (비밀번호)”를 입력받았을 때 입력한 아이디가 존재하지 않으면 “존재하지 않는 아이디입니다” 가 출력되고 (1)로 돌아간다.



```
환영합니다. login이나 register을 통해 로그인이나 회원가입을 해주세요
>login konkuk kk123
존재하지 않는 아이디입니다
login이나 register을 통해 로그인이나 회원가입을 해주세요
```

그림 2. 존재하지 않는 아이디 예시

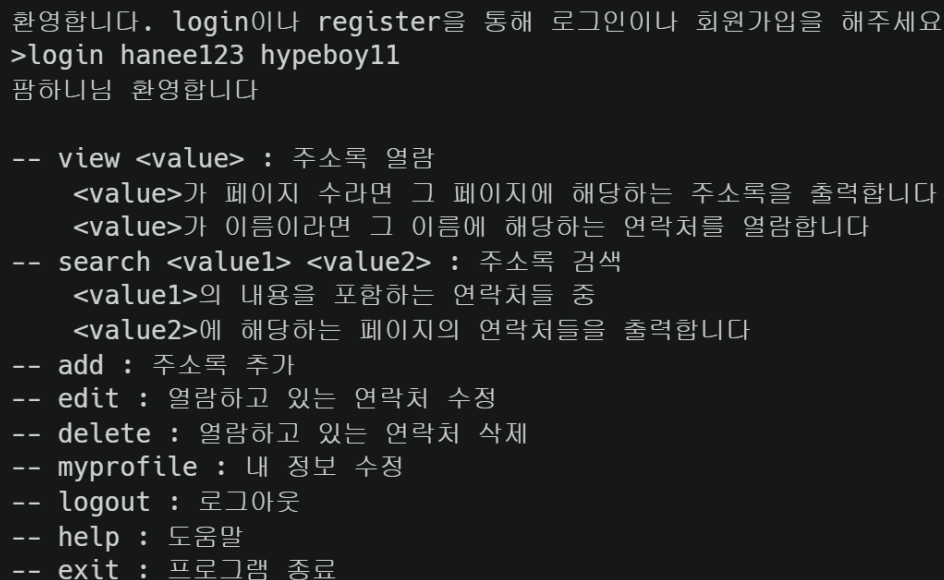
- 입력한 아이디가 존재한다면 비밀번호를 체크하는데, 아이디에 해당하는 비밀번호와 일치하지 않는다면 번호가 틀립니다”가 출력되고 (1)로 돌아간다.



```
환영합니다. login이나 register을 통해 로그인이나 회원가입을 해주세요
>login konkuk1 kk123
비밀번호가 틀립니다
login이나 register을 통해 로그인이나 회원가입을 해주세요
```

그림 3. 비밀번호 오류 예시

- 입력한 비밀번호가 아이디에 해당하는 비밀번호와 일치 한다면 “(사용자 이름)님 환영합니다”가 출력되고 도움말이 출력된다.



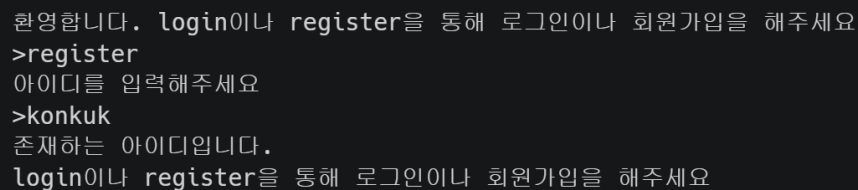
```
환영합니다. login이나 register을 통해 로그인이나 회원가입을 해주세요
>login hanee123 hypeboy11
팜하니님 환영합니다

-- view <value> : 주소록 열람
    <value>가 페이지 수라면 그 페이지에 해당하는 주소록을 출력합니다
    <value>가 이름이라면 그 이름에 해당하는 연락처를 열람합니다
-- search <value1> <value2> : 주소록 검색
    <value1>의 내용을 포함하는 연락처들 중
    <value2>에 해당하는 페이지의 연락처들을 출력합니다
-- add : 주소록 추가
-- edit : 열람하고 있는 연락처 수정
-- delete : 열람하고 있는 연락처 삭제
-- myprofile : 내 정보 수정
-- logout : 로그아웃
-- help : 도움말
-- exit : 프로그램 종료
```

그림 4. 로그인 성공 예시

#### 4.2.1.2 회원가입

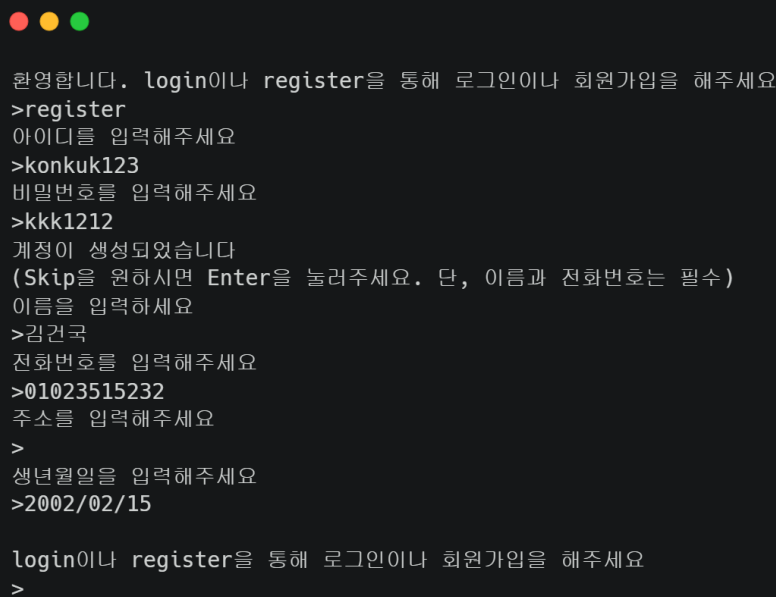
- “register”를 입력받으면 “아이디를 입력해주세요”가 출력되며 아이디 입력을 받는다.
- 입력한 아이디가 이미 존재하면 “존재하는 아이디입니다.”가 출력되고 (1)로 돌아간다.



```
환영합니다. login이나 register을 통해 로그인이나 회원가입을 해주세요
>register
아이디를 입력해주세요
>konkuk
존재하는 아이디입니다.
login이나 register을 통해 로그인이나 회원가입을 해주세요
```

그림 5. 아이디 중복 예시

- 중복된 아이디가 아니라면 “비밀번호를 입력해주세요”가 출력된다.
- 비밀번호를 입력 받은 후 “계정이 생성되었습니다”가 출력되고 사용자 정보를 입력받는다.
- “이름을 입력해주세요”가 출력되고 이름을 입력받는다.
- “전화번호를 입력해주세요”가 출력되고 전화번호를 입력받는다.
- “주소를 입력해주세요”가 출력되고 주소를 입력받는다.
- “생년월일을 입력해주세요 (0000/00/00)”가 출력되고 생일을 입력받는다.
- 모든 입력이 끝난 후 (1)로 돌아간다.



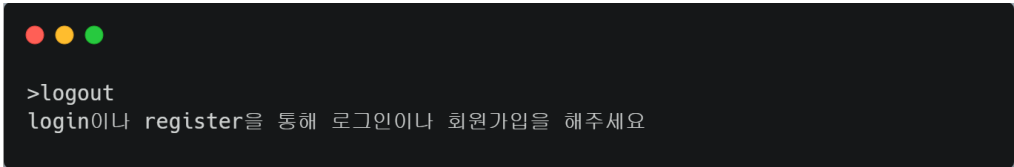
```
환영합니다. login이나 register을 통해 로그인이나 회원가입을 해주세요
>register
아이디를 입력해주세요
>konkuk123
비밀번호를 입력해주세요
>kkk1212
계정이 생성되었습니다
(Skip을 원하시면 Enter을 눌러주세요. 단, 이름과 전화번호는 필수)
이름을 입력하세요
>김건국
전화번호를 입력해주세요
>01023515232
주소를 입력해주세요
>
생년월일을 입력해주세요
>2002/02/15

login이나 register을 통해 로그인이나 회원가입을 해주세요
>
```

그림 6. 회원가입 성공 예시

#### 4.2.1.3 로그아웃

- “logout”을 입력받으면 로그아웃이 되고 (1)로 돌아간다.



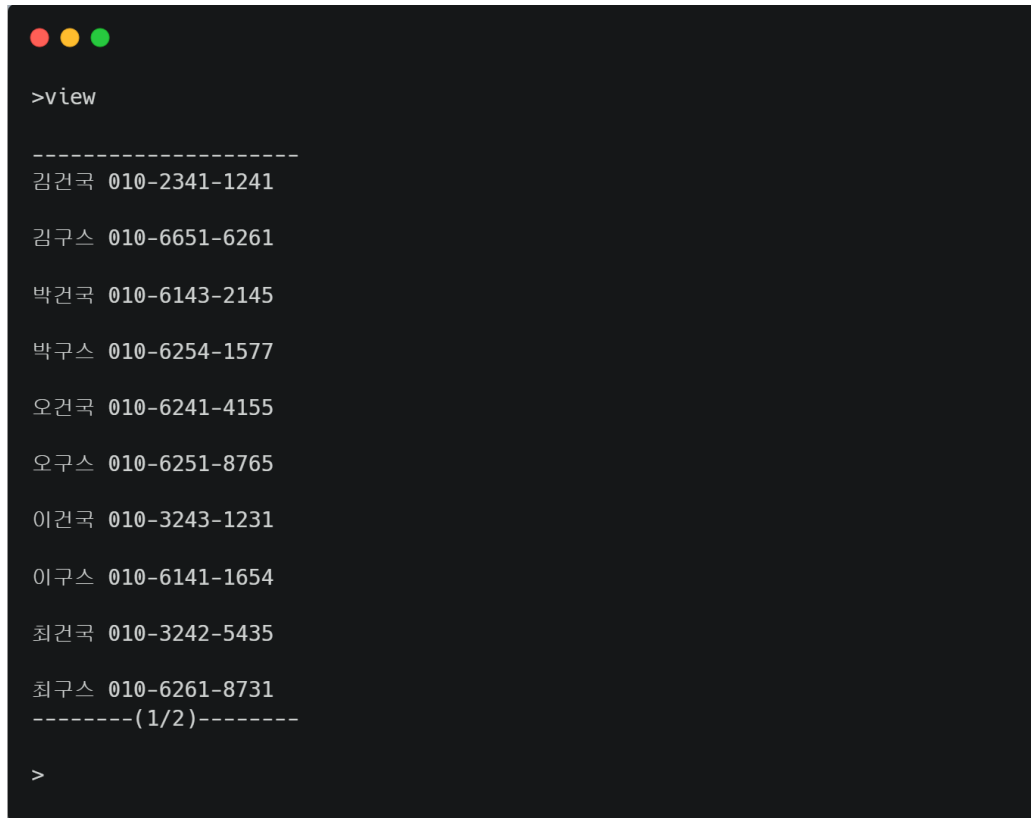
```
>logout
login이나 register을 통해 로그인이나 회원가입을 해주세요
```

그림 7. 로그아웃 예시

#### 4.2.2 주소록 열람

- “view <value>”를 입력받으면 사용자의 주소록을 열람한다.

- 사용자의 연락처들의 개수가 10개가 넘는다면, 10개를 기준으로 나눠 페이지가 생성된다.
- <value>가 공백이라면, 1페이지의 주소록을 출력해준다. 주소록은 기본적으로 이름의 사전 순서로 정렬된다.



```
>view

-----
김건국 010-2341-1241

김구스 010-6651-6261

박건국 010-6143-2145

박구스 010-6254-1577

오건국 010-6241-4155

오구스 010-6251-8765

이건국 010-3243-1231

이구스 010-6141-1654

최건국 010-3242-5435

최구스 010-6261-8731
----- (1/2) -----

>
```

그림 8. 주소록 열람 예시

- <value>가 정수형 데이터라면, 입력받은 페이지의 주소록을 출력해준다.



```
>view 2

-----
김건덕 010-6094-3511

최건덕 010-7893-8854
----- (2/2) -----

>
```

그림 9. 주소록 2페이지 열람 예시

- 입력받은 정수의 페이지가 존재하지 않는다면, “존재하지 않는 페이지입니다”를 출력한다.



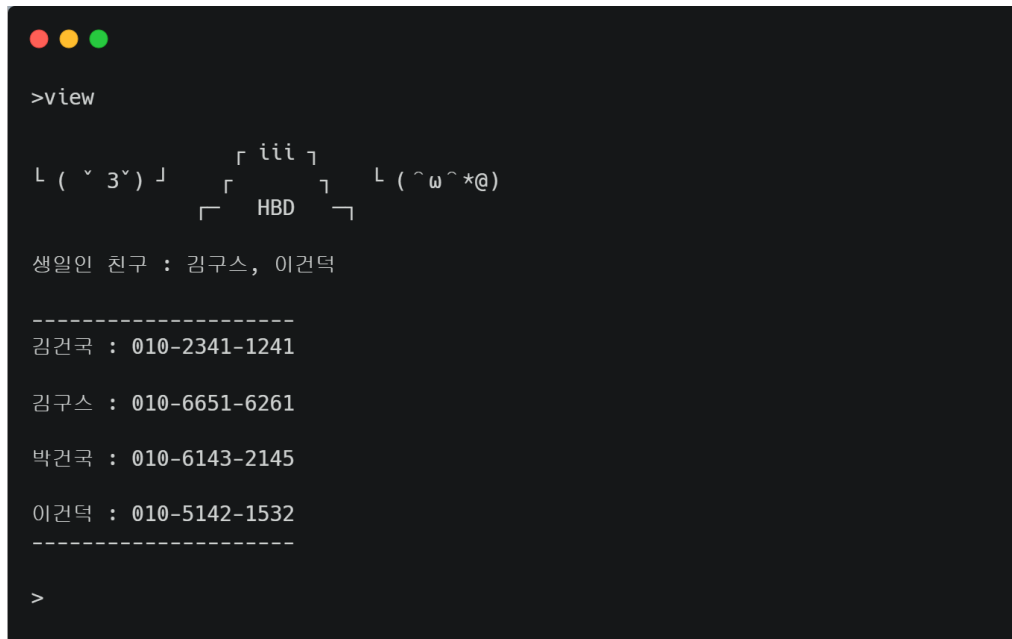
```

>view 3
존재하지 않는 페이지입니다
>

```

그림 10. 존재하지 않는 주소록 페이지 열람 예시

- 당시 날짜가 생일인 연락처가 있을 경우 주소록 1페이지를 출력할 때 생일인 연락처를 출력해준다.



```

>view

      r  iii  r
L ( ^ 3^ ) J  r  HBD  r  L ( ^ω^*@)

생일인 친구 : 김구스, 이건덕

-----
김건국 : 010-2341-1241

김구스 : 010-6651-6261

박건국 : 010-6143-2145

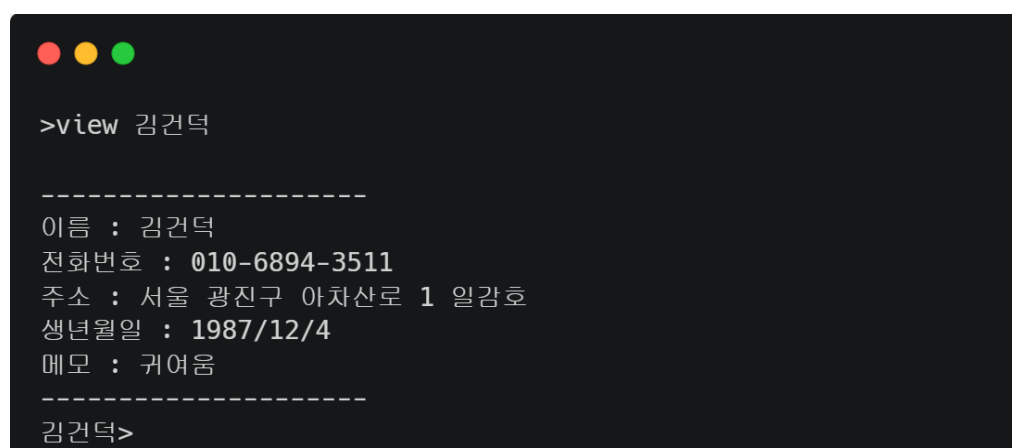
이건덕 : 010-5142-1532
-----

>

```

그림 11. 생일자 출력 예시

- <value>가 문자형 데이터라면 입력받은 문자열의 이름을 가진 연락처를 출력해주고 연락처 열람 상태에 들어간다. 후에 언급할 사례를 제외하면 다른 기능을 수행할 때 연락처 열람 상태에서 빠져나온다.



```

>view 김건덕

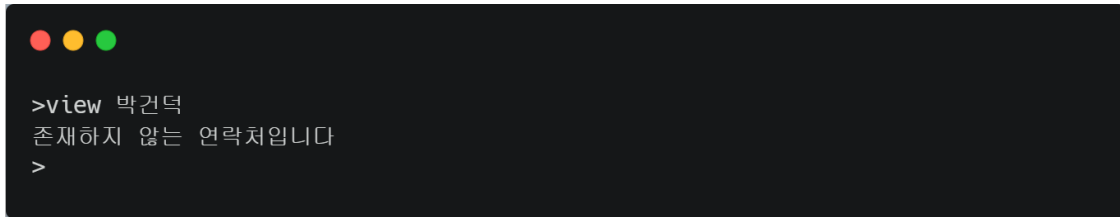
-----
이름 : 김건덕
전화번호 : 010-6894-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----
김건덕>

```

그림 12. 특정 연락처 열람 예시



- 입력받은 문자열의 이름을 가진 연락처가 없다면, “존재하지 않는 연락처입니다”를 출력한다.

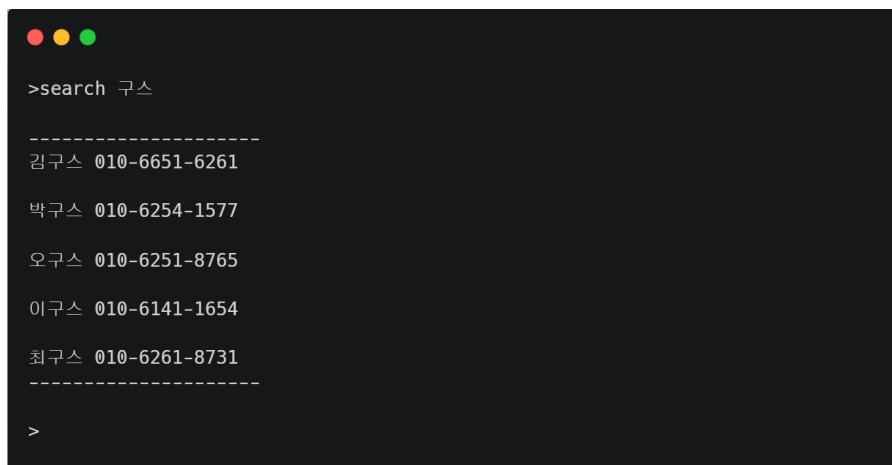


```
>view 박건덕
존재하지 않는 연락처입니다
>
```

그림 13. 존재하지 않는 연락처 검색 예시

### 4.2.3 연락처 검색

- “search <value>”를 입력받으면 연락처 검색을 수행한다.
- 각 연락처들의 모든 요소들을 문자열로 검사해 <value>값을 포함한다면 출력해준다.
- 만약 출력할 연락처들의 개수가 10개가 넘는다면, 주소록 열람과 똑같이 10개를 기준으로 페이지를 생성한다.
- “search <value> <int>” 형식으로 <int>에 해당하는 페이지를 출력하고, <int>가 공백이라면 1페이지를 출력한다.

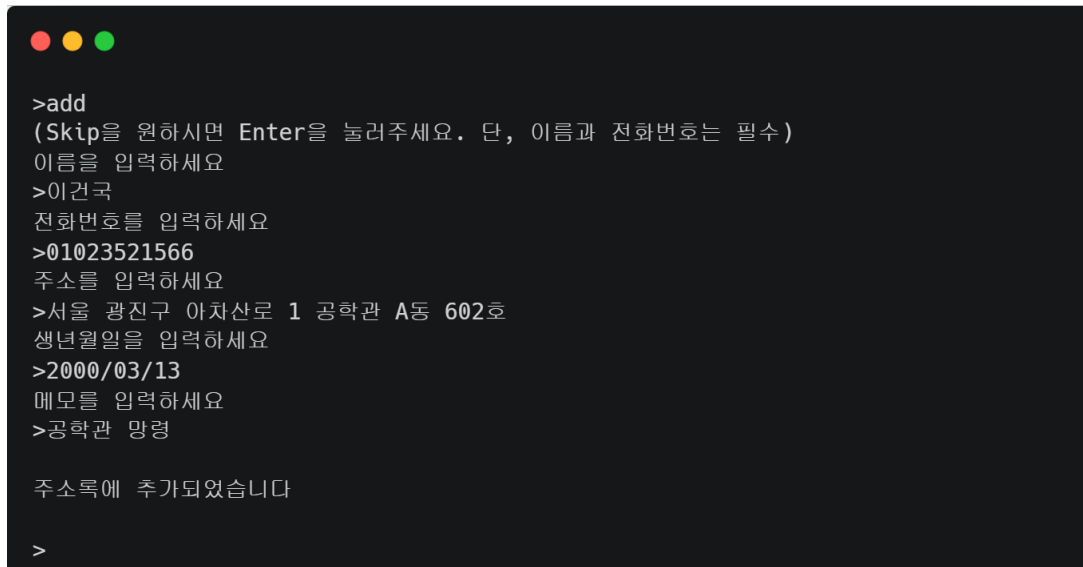


```
>search 구스
-----
김구스 010-6651-6261
박구스 010-6254-1577
오구스 010-6251-8765
이구스 010-6141-1654
최구스 010-6261-8731
-----
>
```

그림 14. “구스”가 포함된 연락처 검색 예시

### 4.2.4 연락처 추가

- “add”를 입력 받으면 연락처를 추가하는 작업을 수행한다.
- 먼저 이름을 입력받고 전화번호를 입력받는다.
- 중복된 전화번호는 4.3 부가확인 목록에서 서술한다.
- 새로운 전화번호라면, 주소와 생년월일, 메모를 입력받는다.



```
>add
(Skip을 원하시면 Enter을 눌러주세요. 단, 이름과 전화번호는 필수)
이름을 입력하세요
>이건국
전화번호를 입력하세요
>01023521566
주소를 입력하세요
>서울 광진구 아차산로 1 공학관 A동 602호
생년월일을 입력하세요
>2000/03/13
메모를 입력하세요
>공학관 망령

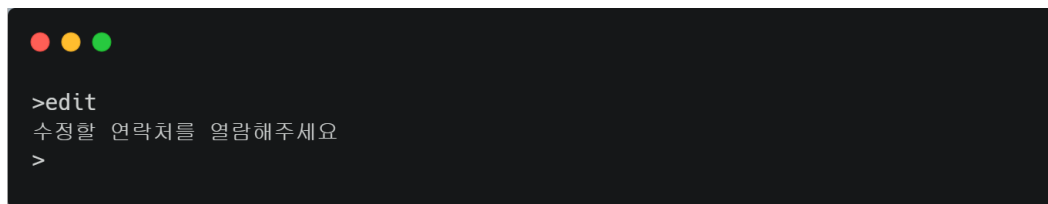
주소록에 추가되었습니다

>
```

그림 15. 연락처 추가 예시

#### 4.2.5 연락처 수정

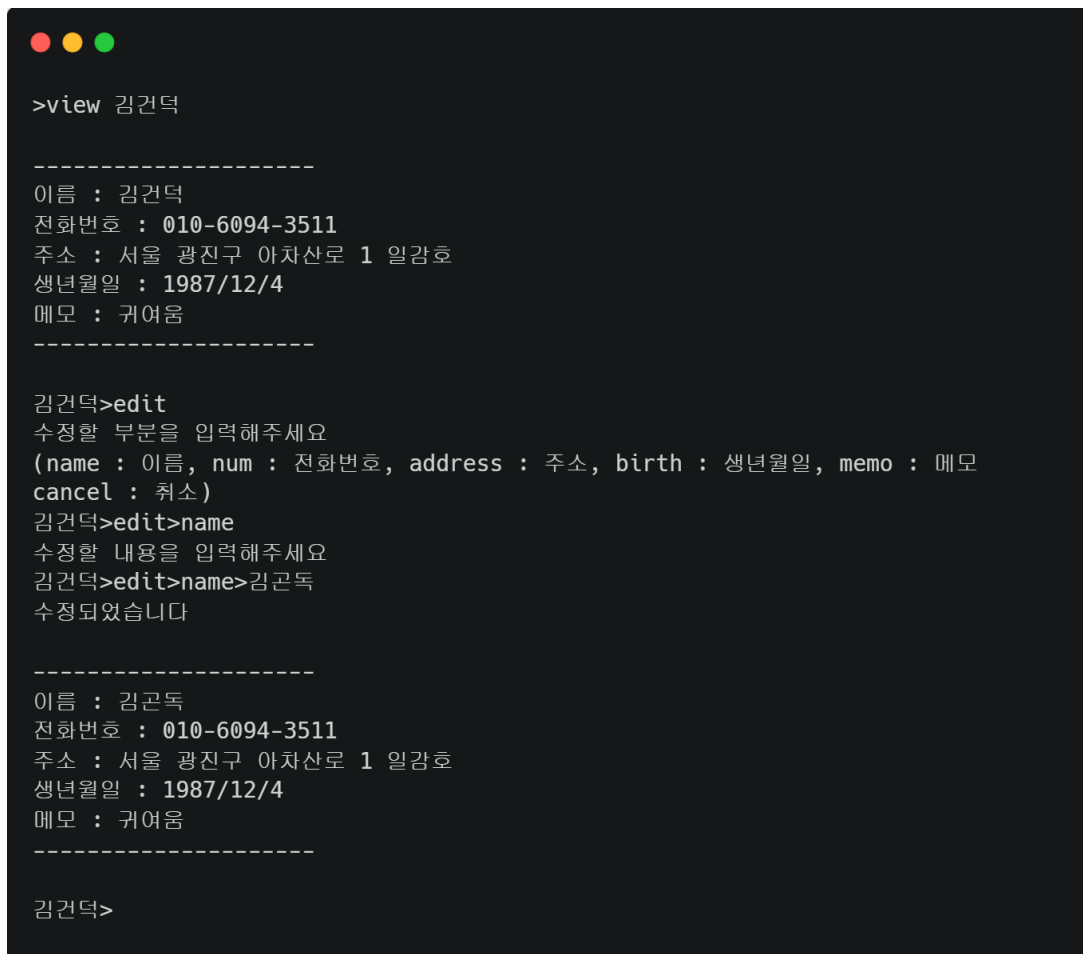
- 연락처 열람 상태가 아닐 때 “edit”을 입력하면 “수정할 연락처를 열람해주세요”를 출력한다



```
>edit
수정할 연락처를 열람해주세요
>
```

그림 16. 연락처 열람 상태가 아닐 때 “edit” 입력 예시

- “view <String>”으로 해당 이름을 가진 연락처를 열람하고 있을 때, “edit”을 입력하면 연락처 수정 기능을 수행한다.
- 수정할 부분을 입력받는데, 입력받은 값이 name이라면 이름을, num이라면 전화번호를, address라면 주소를, birth라면 생년월일을, memo라면 메모를 다시 입력 받아 수정해준다.
- cancel을 입력 받을 때까지 수정을 반복한다.
- 연락처 수정이 끝나도 연락처 열람 상태를 유지한다.



```

>view 김건덕

-----
이름 : 김건덕
전화번호 : 010-6094-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----

김건덕>edit
수정할 부분을 입력해주세요
(name : 이름, num : 전화번호, address : 주소, birth : 생년월일, memo : 메모
cancel : 취소)
김건덕>edit>name
수정할 내용을 입력해주세요
김건덕>edit>name>김곤독
수정되었습니다

-----
이름 : 김곤독
전화번호 : 010-6094-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----

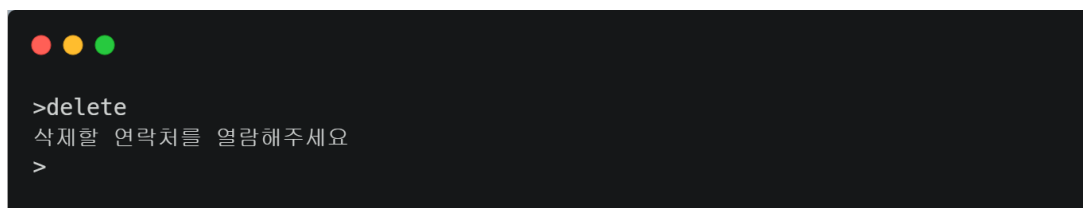
김건덕>

```

그림 17. 연락처 수정 예시

#### 4.2.6 연락처 삭제

- 연락처 열람 상태가 아닐 때 “delete”를 입력하면 “삭제할 연락처를 열람해주세요”를 출력한다.



```

>delete
삭제할 연락처를 열람해주세요
>

```

그림 18. 연락처 열람 상태가 아닐 때 “delete” 입력 예시

- “view <String>”으로 해당 이름을 가진 연락처를 열람하고 있을 때, “delete”를 입력하면 연락처 삭제 기능을 수행한다.
- 삭제를 하기 전 “정말 삭제하시겠습니까? 삭제를 원하시면 “삭제”를 입력해주세요”를 출력하고 문자열을 입력받는다.
- “삭제”를 입력받으면 삭제를 진행한다.

```

>view 김건덕

-----
이름 : 김건덕
전화번호 : 010-6094-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----

김건덕>delete
정말 삭제하시겠습니까? 삭제를 원하시면 "삭제"를 입력해주세요
김건덕>delete>삭제
삭제되었습니다

>

```

그림 19. 연락처 삭제 예시

- “삭제”를 입력받지 않으면 해당 연락처를 삭제 하지 않고 연락처 열람 상태를 유지한다.

```

>view 김건덕

-----
이름 : 김건덕
전화번호 : 010-6094-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----

김건덕>delete
정말 삭제하시겠습니까? 삭제를 원하시면 "삭제"를 입력해주세요
김건덕>삭제>ㄴㄴ
삭제되지 않았습니다

김건덕>

```

그림 20. 삭제 취소 예시

#### 4.2.7 내 정보 수정

- “myprofile”을 입력 받으면 내 정보를 출력하고 수정할 부분을 입력받는다.
- 입력받은 부분이 name이라면 이름을, num이라면 전화번호를, address라면 주소를, birth라면 생년월일을 다시 입력받아 수정해준다.
- cancel을 입력 받을 때까지 수정을 반복한다.
- 수정 후 수정된 내 정보를 출력해준다.

```

>myprofile

-----내 정보-----
이름 : 김건국
전화번호 : 010-1313-3232
주소 : 서울 광진구 아차산로 1
생년월일 : 2002/02/15
-----

수정할 부분을 입력해주세요
(name : 이름, num : 전화번호, address : 주소, birth : 생년월일, cancel : 취소)
내 정보 수정>num
수정할 내용을 입력해주세요
내 정보 수정>전화번호>010-3253-2532

-----내 정보-----
이름 : 김건국
전화번호 : 010-3253-2532
주소 : 서울 광진구 아차산로 1
생년월일 : 2002/02/15
-----

>

```

그림 21. 내 정보 수정 예시

#### 4.2.8 도움말

- “help”를 입력받으면 사용자가 입력할 수 있는 명령어들과 그에 대한 설명을 출력해준다.

```

>help

-- view <value> : 주소록 열람
    <value>가 페이지 수라면 그 페이지에 해당하는 주소록을 출력합니다
    <value>가 이름이라면 그 이름에 해당하는 연락처를 열람합니다
-- search <value1> <value2> : 주소록 검색
    <value1>의 내용을 포함하는 연락처들 중
    <value2>에 해당하는 페이지의 연락처들을 출력합니다
-- add : 주소록 추가
-- edit : 열람하고 있는 연락처 수정
-- delete : 열람하고 있는 연락처 삭제
-- myprofile : 내 정보 수정
-- logout : 로그아웃
-- help : 도움말
-- exit : 프로그램 종료

>

```

그림 22. 도움말 예시

#### 4.2.9 종료

- “exit”를 입력하면 프로그램이 종료된다.

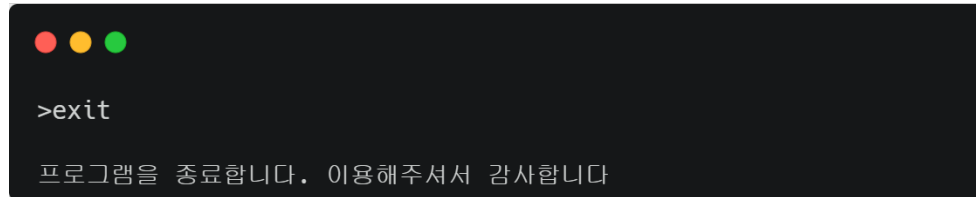


그림 23. 프로그램 종료 예시

## 5 데이터 파일

- 기본적으로, 이 프로그램은 실행 중에 사용자로부터 연락처 정보를 키 입력으로 받아들이고 이 정보들을 프로그램 스스로 내부 기능을 통해 데이터 파일에 저장한다.
- 모든 사용자의 아이디, 비밀번호 및 개인정보를 담은 File 하나와 각 유저의 개인별 연락처를 담은 개인별 연락처 File로 구성된다.
- 사용자가 별도의 텍스트 편집기를 이용해 데이터 파일을 (프로그램을 통하지 않고) 직접 생성/편집/저장하는 방식도 지원한다. 단:
  - 프로그램이 실행되고 있는 중 데이터 파일을 직접 편집할 경우, 정상 작동을 보장하지 않는다.
  - 데이터 파일은 (BOM 없는) UTF-8 인코딩으로 저장되어야 한다.
  - 운영체제마다 개행 문자들을 다루는 기본 방식이 서로 다르지만(LF, CRLF 등) 이 프로그램은 모든 개행 방식을 똑같이 다루므로, 어떤 OS에서 어떤 방식으로 저장하여도 상관 없다.
  - 이 절에서 정하는 문법과 의미 규칙을 준수하며 편집해야만 프로그램이 정상적으로 동작한다.
- 사용자는 문법에만 맞다면 얼마든지 데이터 파일을 수정해도 되지만, 데이터의 무결성을 보장할 수 없으므로 권장하지는 않는다.
- 위 마지막 항목에 의거하여, 사용자가 데이터 파일을 올바르게 편집하고 어떤 상황에서 어떤 결과가 나올지 정확히 예상할 수 있도록 데이터 파일의 문법(형식)과 의미 규칙(추가 조건)들을 명시한다.

### 5.1 문법 규칙

#### 5.1.1 기본 규칙

- 기본적으로 데이터 파일은 Tab Delimited Value 형식으로 작성되며, Column은 Tab문자<sup>U+0009</sup> 1개로, Row는 개행으로 구분한다.
- 각각의 Row는 각 Field와 Tab문자를 번갈아가며 입력되며, Field는 비어있을 수 있다.
- 위 경우 Tab문자가 두 번 이상 등장할 수 있다.

### 5.1.2 회원 정보 데이터 파일 규칙 및 예시

회원 정보 데이터 파일에 저장될 Field들은 다음과 같다.

- 아이디<sup>필수 항목</sup>
- 비밀번호<sup>필수 항목</sup>
- 이름<sup>필수 항목</sup>
- 전화번호<sup>필수 항목</sup>
- 주소<sup>선택 항목</sup>
- 생년월일<sup>선택 항목</sup>

각각의 항목들은 위 순서대로 Tab문자(이하 예시에서는 Tab문자를 시각적으로 표현하기 위해 ‘→’로 표기)로 구분되어 한 줄로 구성되어야 한다. 선택 항목을 공란으로 비워두는 경우, Tab 문자를 한번 더 입력한 후 다음 항목을 바로 입력한다.

예를 들어, 아이디가 ‘sterdsterd’이고, 비밀번호가 ‘1q2w3e4r’이고, 이름이 ‘이율원’이고, 전화번호가 ‘010-8496-3744’이고, 주소가 ‘광진구 능동로 120’이고, 생년월일이 ‘20030612’인 사람의 정보는 다음과 같이 표현할 수 있다.

**sterdsterd → 1q2w3e4r → 이율원 → 010-8496-3744 → 광진구 능동로 120 → 20030612**

처럼 표현한다. 또한, 아이디가 ‘kucse’이고, 비밀번호가 ‘1234’이고, 이름이 ‘김건کم’이고, 전화번호가 ‘01012345678’이고, 주소와 생년월일이 생략되어 있는 사람의 정보는 다음과 같이 표현할 수 있다,

**kucse → 1234 → 김건کم → 01012345678 → →**

### 5.1.3 개인별 데이터 파일 규칙 및 예시

개인별 데이터 파일에 저장될 Field들은 다음과 같다.

- 이름<sup>필수 항목</sup>
- 전화번호<sup>필수 항목</sup>
- 주소<sup>선택 항목</sup>
- 생일<sup>선택 항목</sup>
- 메모<sup>선택 항목</sup>

각각의 항목들은 위 순서대로 Tab문자(이하 예시에서는 Tab문자를 시각적으로 표현하기 위해 ‘→’로 표기)로 구분되어 한 줄로 구성되어야 한다. 단, 메모에는 Tab문자가 포함될 수 있기 때문에, 가장 처음 4개의 Tab문자를 기준으로 하여 각 항목을 구분한다. 선택 항목을 공란으로 비워두는 경우, Tab 문자를 한번 더 입력한 후 다음 항목을 바로 입력한다.

예를 들어, 이름이 ‘이율원’이고, 전화번호가 ‘010-8496-3744’이고, 주소가 ‘광진구 능동로 120’이고, 생일이 ‘20030612’이고, 메모가 ‘کم공22 비빔밥 미래인재양성부 장관’인 사람의 정보는 다음과 같이 표현할 수 있다.

이율원 → 010-8496-3744 → 광진구 능동로 120 → 20030612 → کم공 22 비빔밥 미래인재양성부 장관

처럼 표현한다. 또한, 아이디가 ‘kucse’이고, 비밀번호가 ‘1234’이고, 이름이 ‘김건کم’이고, 전화번호가 ‘01012345678’이고, 주소와 생년월일이 생략되어 있고, 메모가 ‘کم공 → 2과대’인 사람의 정보는 다음과 같이 표현할 수 있다,

kucse → 1234 → 김건کم → 01012345678 → → → کم공 → 2과대

## 5.2 의미 규칙

- 데이터 파일 속의 서로 다른 두 Row들 속에는 서로 같은 ‘전화번호’가 존재할 수 없다

## 5.3 부가 확인 목록

- 동일한 이름이 둘 이상의 Row에 속하는 경우,
- 동명이인이 있는 경우 자동적으로 추가된 연락처 이름 뒤에 (순서)가 붙는다.
- 예를 들어, 김건국이 있는 상태에서 김건국 추가 시 김건국(1)이 된다.
- 또한, 김건국과 김건국(1)이 있는 상태에서 김건국 추가 시 김건국(2)가 된다.

## 5.4 무결성 확인 및 처리

- 프로그램 시작 시 파일을 읽어 주소록 객체를 생성하는데 실패 한다면 실패 메시지를 출력하고 종료한다.

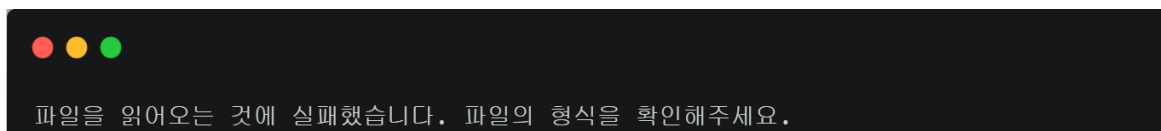
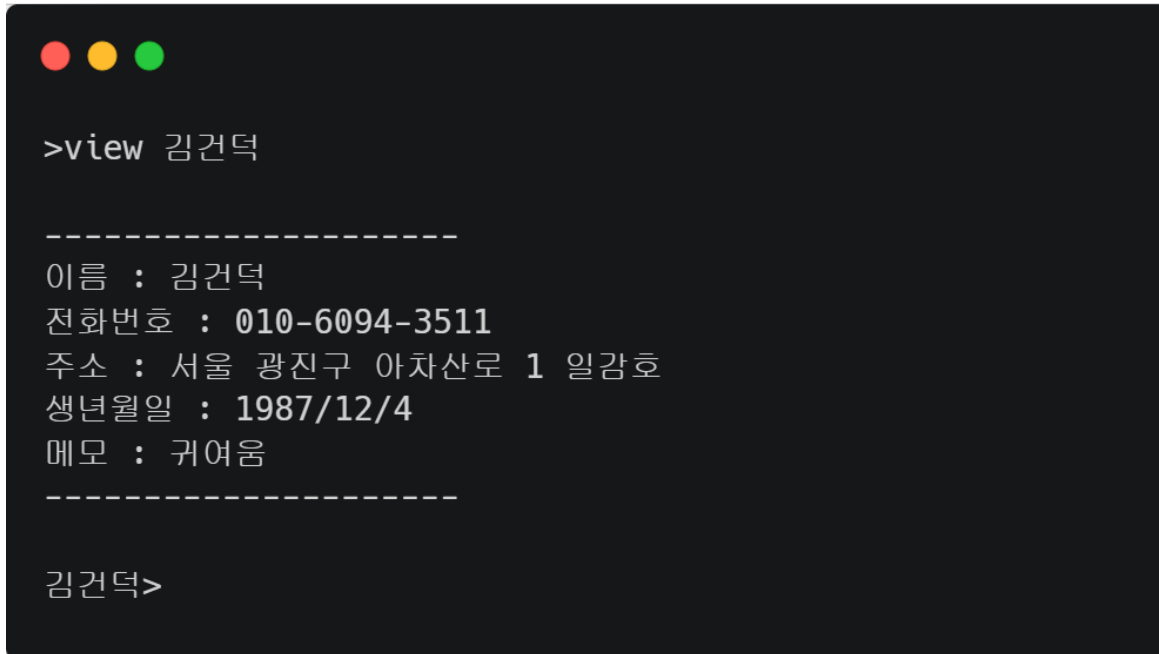


그림 24. 파일 읽기 오류 예시



## 6 주 프롬프트

- 주 프롬프트는 화면에 다음과 같이 현재 선택한 메뉴 계층을 출력하고, 사용자의 키 입력을 기다린다.



```

>view 김건덕

-----
이름 : 김건덕
전화번호 : 010-6094-3511
주소 : 서울 광진구 아차산로 1 일감호
생년월일 : 1987/12/4
메모 : 귀여움
-----

김건덕>

```

그림 25. 주 프롬프트 출력 예시

- 문법 형식:** 주 프롬프트에 키 입력하는 올바른 문법은 다음 형식과 같다:
  - <명령어><공백><단어열>
  - 단, 위의 <명령어> 자리에는 반드시 다음 표의 단어들 중 하나를 써야하며, 이들만이 문법적으로 올바른 '명령어'다.

login	register	logout	view	search	add
edit	delete	myprofile	help	exit	

표 1. 올바른 모든 명령어들

- 해석:** 명령어는 이미 문법 형식에 구분되어 쓰여있으므로 해석 규칙이 필요 없다. <단어열>은 (만일 있다면) 그 속의 단어들 각각을 인자들로 간주한다.
- 의미 규칙:** <명령어> 자리에 어떤 명령어 군에 속한 명령어가 들어오는지에 따라서, <단어열> 중 인식할 인자의 개수가 달라진다. 인자들은 공백으로 구분되며 만일 인자들의 개수가 인식할 인자보다 많다면, 초과된 인자들은 전부 무시한다.

```

>help 도움말 보려면 어디로 가야돼요? 뉴진스의 하입보이요

-- view <value> : 주소록 열람
    <value>가 페이지 수라면 그 페이지에 해당하는 주소록을 출력합니다
    <value>가 이름이라면 그 이름에 해당하는 연락처를 열람합니다
-- search <value1> <value2> : 주소록 검색
    <value1>의 내용을 포함하는 연락처들 중
    <value2>에 해당하는 페이지의 연락처들을 출력합니다
-- add : 주소록 추가
-- edit : 열람하고 있는 연락처 수정
-- delete : 열람하고 있는 연락처 삭제
-- myprofile : 내 정보 수정
-- logout : 로그아웃
-- help : 도움말
-- exit : 프로그램 종료

>

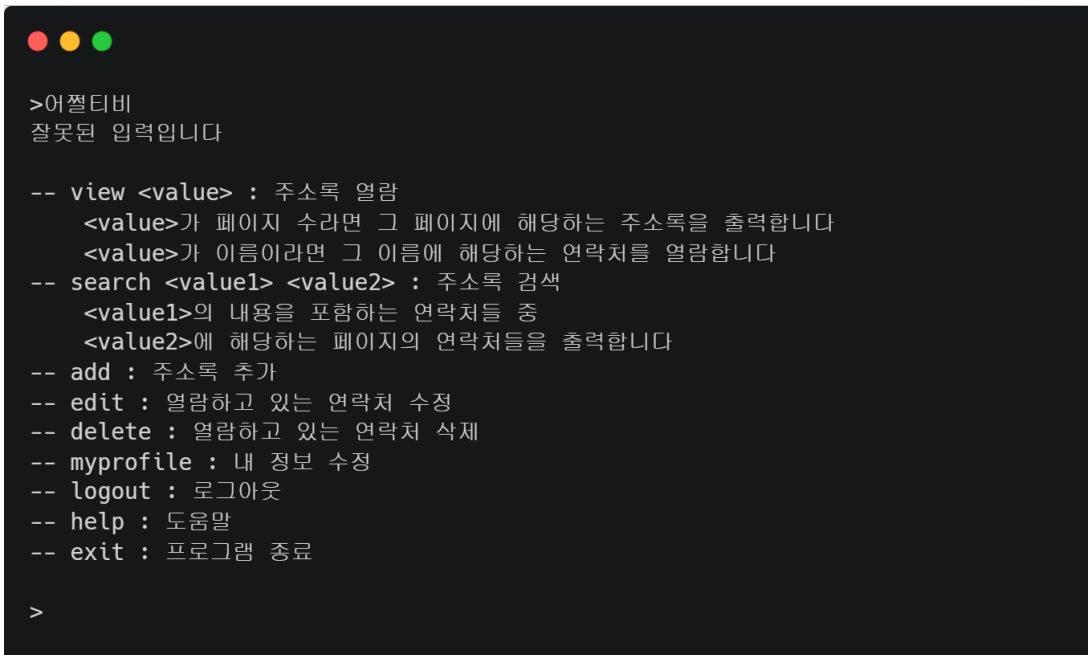
```

그림 26. 초과된 인자 예시

명령어	인자 개수	설명
login	2개	데이터 파일에서 인자들을 비교해 로그인을 진행
register	없음	회원가입을 진행
logout	없음	로그인/회원가입 창으로 돌아감
view	없거나, 1개	인자에 따라 연락처 열람을 진행
search	1개 또는 2개	데이터 파일에서 인자를 검색
add	없음	주소록에 연락처를 추가
edit	없음	열람하고 있는 연락처를 수정
delete	없음	열람하고 있는 연락처를 삭제
myprofile	없음	내 정보를 열람 및 수정
help	없음	프로그램에서 사용가능한 명령어 목록 출력
exit	없음	프로그램 종료

표 2. 명령어들과 인자 규칙 및 설명

- **문법 오류 결과:** 만일 주 프롬프트에서 입력한 첫번째 단어가 명령어가 아닐 경우 잘못된 입력으로 간주하고 도움말을 출력해준다.



```

>어쩔티비
잘못된 입력입니다

-- view <value> : 주소록 열람
  <value>가 페이지 수라면 그 페이지에 해당하는 주소록을 출력합니다
  <value>가 이름이라면 그 이름에 해당하는 연락처를 열람합니다
-- search <value1> <value2> : 주소록 검색
  <value1>의 내용을 포함하는 연락처들 중
  <value2>에 해당하는 페이지의 연락처들을 출력합니다
-- add : 주소록 추가
-- edit : 열람하고 있는 연락처 수정
-- delete : 열람하고 있는 연락처 삭제
-- myprofile : 내 정보 수정
-- logout : 로그아웃
-- help : 도움말
-- exit : 프로그램 종료

>

```

그림 27. 잘못된 입력 예시

- **의미 오류 혹은 정상 결과:** 만일 입력 중 첫번째 단어가 명령어가 맞으면 명령어별 기능을 수행한다. 명령어 별 기능과 정상결과는 [4.2절](#)에서 서술했으며, 아래에서는 명령어 별 인자 문법 형식과 비정상 결과에 대해 서술하겠다.

## 6.1 도움말

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 비정상 결과는 존재하지 않는다.

## 6.2 종료

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 비정상 결과는 존재하지 않는다.

## 6.3 로그인 & 가입 명령어 군

### 6.3.1 로그인

- **인자 문법 형식:** <아이디><공백><비밀번호> 순서로 입력받는다. 앞의 두개 인자를 제외한 추가적인 인자들이 있다면 무시한다.

- **비정상 결과:** 인자의 수가 부족하다면 “입력 오류”를 출력한다.

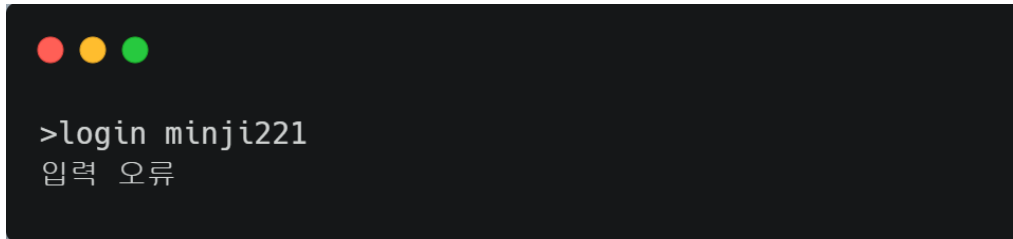


그림 28. 로그인 입력 오류 예시

### 6.3.2 회원가입

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 비정상 결과는 존재하지 않는다.

### 6.3.3 로그아웃

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 비정상 결과는 존재하지 않는다.

## 6.4 열람

- 1개보다 많은 인자가 입력 되면, 앞의 1개를 제외한 나머지 인자들은 무시한다.
- **인자 문법 형식 1:** 인자가 없다면, 사용자의 주소록의 1페이지를 열람한다.
- **인자 문법 형식 2:** 인자가 정수라면, 인자에 해당하는 페이지의 주소록을 열람한다.
- **인자 문법 형식 3:** 인자가 문자열이라면, 인자에 해당하는 이름을 가진 연락처를 열람한다.
- **비정상 결과 1:** 인자가 정수일 때, 인자에 해당하는 페이지가 존재하지 않는다면, 오류 메시지를 출력한다.
- **비정상 결과 2:** 인자가 문자열일 때, 인자에 해당하는 이름을 가진 연락처가 존재하지 않는다면, 오류 메시지를 출력한다.

## 6.5 추가

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 비정상 결과는 존재하지 않는다.

## 6.6 삭제

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 4.2.6절에서 서술했듯이 연락처 열람 상태가 아니라면 “삭제할 연락처를 열람해주세요”를 출력한다.

## 6.7 수정

- **인자 문법 형식:** 인자가 있더라도 전부 무시하고 같은 기능을 수행한다.
- **비정상 결과:** 4.2.5절에서 서술했듯이 연락처 열람 상태가 아니라면 “수정할 연락처를 열람해주세요”를 출력한다.

## 6.8 검색

- **인자 문법 형식:** 인자를 문자열로 1개만 입력 받고, 그 뒤의 인자들은 무시한다.
- **비정상 결과:** 인자가 없다면, “입력 오류”를 출력한다.

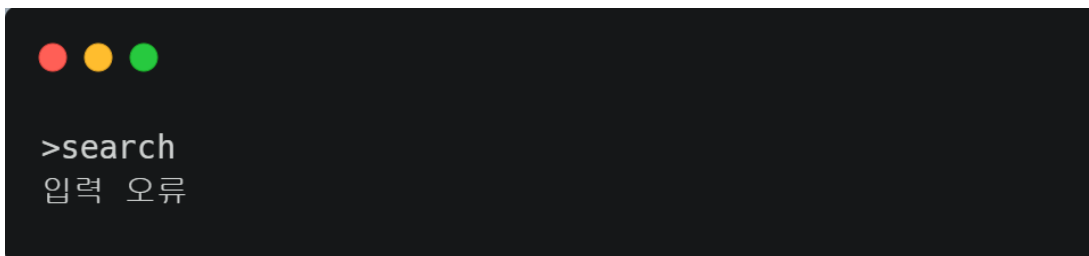


그림 29. 검색 입력 오류 예시

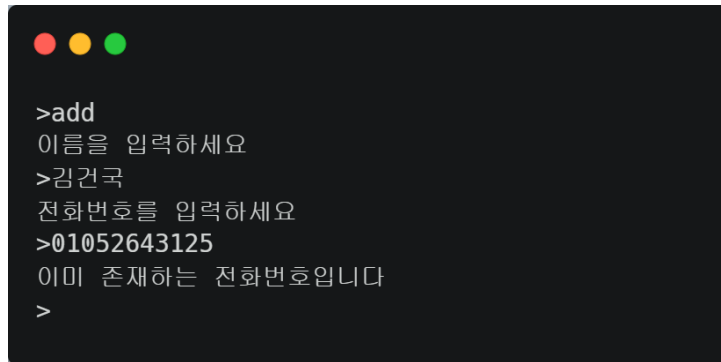
## 7 예외 처리

### 7.1 동명이인

- 동명이인이 있는 경우 자동적으로 추가된 연락처 이름 뒤에 (순서)가 붙는다.
- ex) 김건국이 있는 상태에서 김건국 추가 시 김건국(1)이 된다.

### 7.2 중복 전화번호

- 만약 주소록에 같은 전화번호를 가진 연락처가 존재한다면, “이미 존재하는 전화번호입니다”를 출력한다.



```
>add
이름을 입력하세요
>김건국
전화번호를 입력하세요
>01052643125
이미 존재하는 전화번호입니다
>
```

그림 30. 중복 전화번호 입력 예시

### 7.3 이름이 정수(Integer)

- 연락처 추가에서 이름을 정수로 입력한다면 “정수로 이뤄진 이름은 불가능합니다”를 출력하고 이름을 다시 입력받는다.

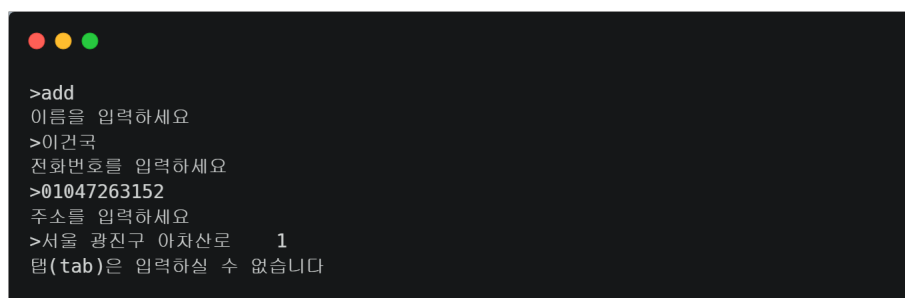


```
>add
이름을 입력하세요
>1
정수로 이뤄진 이름은 불가능합니다
```

그림 31. 정수로 이뤄진 이름 입력 예시

### 7.4 주소록 요소들에 탭 문자(\t) 존재

- 프로그램에서 데이터 파일을 저장하고 불러올 때, 요소들을 나누는 기준이 탭 이므로, 연락처를 추가, 수정, 내 정보 수정에서 오류를 발생 시킬 수 있는 탭을 입력 받으면 “탭(tab)은 입력하실 수 없습니다”를 출력하고 해당 요소를 다시 입력받는다.

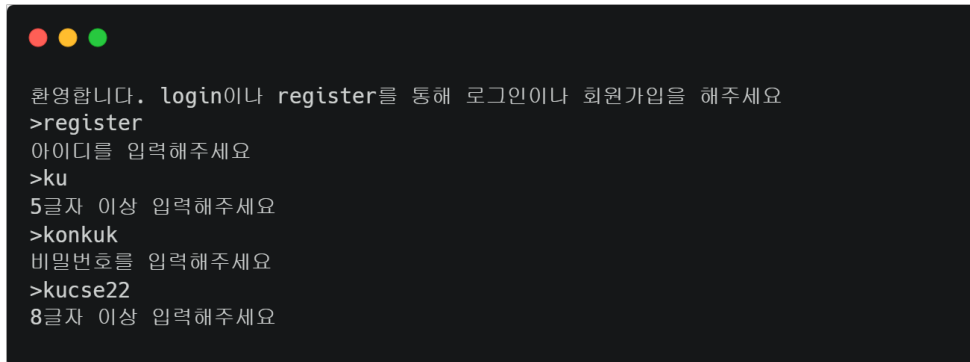


```
>add
이름을 입력하세요
>이건국
전화번호를 입력하세요
>01047263152
주소를 입력하세요
>서울 광진구 아차산로 1
탭(tab)은 입력하실 수 없습니다
```

그림 32. 탭 문자(\t) 입력 예시

## 7.5 문법 오류

- [4.1절](#)에 정의한 요소들의 문법 형식에 맞지 않게 입력하였을 경우, 각 문법 형식에 대한 안내를 출력하며 해당 요소를 다시 입력받는다.



```
환영합니다. login이나 register를 통해 로그인이나 회원가입을 해주세요
>register
아이디를 입력해주세요
>ku
5글자 이상 입력해주세요
>konkuk
비밀번호를 입력해주세요
>kucse22
8글자 이상 입력해주세요
```

그림 33. 문법 형식에 맞지 않는 입력 예시



```
환영합니다. login이나 register를 통해 로그인이나 회원가입을 해주세요
>register
아이디를 입력해주세요
>konkuk123
비밀번호를 입력해주세요
>konkukcse22
계정이 생성되었습니다
이름을 입력해주세요
>최강공
전화번호를 입력해주세요
>010-1234-5678
주소를 입력해주세요
>광진구 능동로 1
생년월일을 입력해주세요
>2003/13/42
존재하지 않는 날짜입니다. 다시 입력해주세요
>2003/06/16

login이나 register를 통해 로그인이나 회원가입을 해주세요
>
```

그림 34. 문법 형식에 맞지 않는 입력 예시 2