전공기초프로젝트1
1차 기획서 수정 3판

-가계부 관리 프로그램-

Project Team

A분반 4팀

Team information

201811236 구선민

201611273 유병헌

201811170 김정우

202011363 정세훈

202010655 이윤정

원판과의 차이점

[5.1절] 문자 문자열 조건 일부(연도 자리수) 변경

[5.1절] 입력 조건 일부(순서 상관) 변경

[5.3절] 태그 규칙 조건 일부(문자열 사이에 ']'문자 포함 금지) 추가

[5.4.1절] 주민등록번호 예시 표준 공백 누락 수정

[5.4.2절] 목업에 입력된 비밀번호 예시가 문법 규칙 위배 수정

[7.3절] 의미 규칙 일부(추가 시 태그 의미 규칙 고려) 제거

[7.3.1절] '내역'을 '이름'으로 표현 실수

[7.3.1절] 제시된 오류 인자 2개 이하에 대해 모두 출력하도록 목업 예시 수정

[7.3.2절] 목업 부호 누락 수정

[7.4절] 의미 규칙 일부(추가 시 태그 의미 규칙 고려) 제거

[7.5.1절] 목업 출력 모순(규칙과 맞지 않는 자리수 제시) 수정

[7.7절] 계좌 메뉴 출력 목업 '1.' 출력 누락 수정

목 차

1. 개요 3
2. 용어 3
3. 기본 사항 7
3.1 작동 환경 7
3.2 프로그램 구성 7
3.3 프로그램 설치 및 실행 8
4. 사용 흐름도 9
5. 데이터 요소 10
5.1 날짜 10
5.2 금액 11
5.3태그 12
5.4 회원가입 시 14
5.4.1 회원 정보(이름, 주민등록번호) 14
5.4.2 계정 정보(ID, 비밀번호) 16
6. 데이터 파일 18
6.1 의미 규칙 18
6.2 무결성 확인 및 처리 19
7. 주 프롬프트 22
7.1 종료 명령어군 24
7.2 무결성 확인/처리 명령어군 24
7.3 추가 명령어군 24
7.3.1 부 프롬프트 1 : 내역 입력 26
7.3.2 부 프롬프트 2 : 저장 확인 27
7.4 검색 및 수정 명령어군 28
7.4.1 부 프롬프트 1 : 내역 선택 29
7.4.2 부 프롬프트 2 : 검색 후 작업 선택 29
7.4.3 부 프롬프트 3 : 내역 수정 30
7.4.4 부 프롬프트 4 : 수정 확인 32
7.4.5 부 프롬프트 5 : 내역 삭제 32
7.5 태그 편집 명령어군 32
7.5.1 부 프롬프트 1 : 태그 추가 33
7.5.2 부 프롬프트 2 : 태그 수정 35
7.5.3 부 프롬프트 3 : 태그 삭제 36
7.6 권한 변경 명령어군 37
7.6.1 부 프롬프트 1 : 사용자 이름 입력 37

7.6.2 부 프롬프트 2 : 권한 입력 38
7.6.3 부 프롬프트 3 : 저장 확인 39
7.7 계좌 선택 명령어군 39
7.7.1 부 프롬프트 3 : 계좌 선택 40
7.7.2 부 프롬프트 3 : 계좌 생성(공용 계좌) 40
7.7.3 부 프롬프트 3 : 권한 요청 40

1. 개요

개인 계좌 및 공용 계좌의 가계부를 관리 및 검색할 수 있는 프로그램.

명령어 및 그 인자들을 키 입력 방식으로 사용하며, 텍스트 파일에 데이터를 저장하고, 한 사람은 1개의 개인 계좌 및 다수의 공용 계좌를 가질 수 있으며, 수입/지출을 추가 및 수정(삭제)이 가능하며, 태그 및 구간을 통해 검색할 수 있습니다. 내역 수정 권한을 가진 회원과 관리자만 내역을 수정할 수 있으며, 태그 수정 및 권한 변경은 오직 관리자만 할 수 있습니다.

2. 용어

이 문서에서 사용할 용어들의 의미를 약속하고, 아울러 일부 용어들과 관련하여 이 문서에서 사용할 표기법도 함께 약속해둡니다. 여기서 약속하는 용어들은 다음과 같이 분류됩니다 :

■ 이미 세간에서 지배적으로 통용되는 의미대로지만, 소개나 설명이 필요한 경우 (용어에 기존 표식)

■ 세간에서 지배적으로 통용되는 (포괄적인) 의미에서 벗어나지는 않고 여전히 그에 포함되지만, 보다 특정한 범위나 대상으로 좁혀서 사용하려는 경우 (용어에 특정 표식)

■ 세간에서 둘 이상의 의미들로 혼란스럽게 통용되고 있어서, 그 중 어떤 의미로 사용할지 명확히 정하려는 경우 (용어에 명확 표식)

■ 세간에서 널리 통용되는 의미(들)에서 다소 벗어난 다른 의미로 바꿔 사용하려는 경우 (용어에 변경 표식)

■ 세간에서는 거의 사용되고 있지 않은 용어를 이 문서에서만 쓰려고 새로 정의하는 경우 (용어에 신규 표식)

이 절에서 정의한 용어들은 내용상 비슷한 계열의 관련 용어끼리 가까이 묶되, 묶음 간의 순서는 본문에 등장하는 순서를 가능한 한 따르고 있습니다. 묶음 속의 순서는 만일 A 용어를 이용해서 B 용어를 정의해야 할 경우 A를 B 보다 먼저 정의하는 식으로 되어있습니다.

홈 경로기존 운영체제가 각 사용자 (계정)에게 제공하는 고유의 디렉토리 (폴더)의 경로. 이 문서에서 경로명 전체 혹은 시작 부분에 “{HOME}”이라고 표기된 부분은 항상 홈 경로입니다. 홈 경로는 운영체제 관리자의 설정에 따라 달라질 수 있지만, 만약 사용자 계정명이 “konkuk”이라면 기본적으로,

- MS Windows Vista, 7, 8, 10의 경우 “C:₩Users₩konkuk”입니다.

- Linux (중 대부분의 배포판들) 등 FHS를 준용하는 운영체제의 경우 “/home/konkuk”입니다.
- Apple OS X, macOS의 경우 “/Users/konkuk”입니다.

가계부 관리 프로그램 (혹은 **프로그램**)[특정] 이 문서를 통해 기획 및 명세하고 있는 대상 프로그램.

주 실행 파일 (혹은 **실행 파일**)[특정+변경] 가계부 관리 프로그램을 실행시키기 위해 꼭 필요한 본체 파일. 이 파일은 스스로(혹은 운영체제의 도움만 받아서) 실행 가능한 파일은 아니고, 엄밀히 말하면 Python 인터프리터에 의해 읽히고 해석될 내용을 담고 있는 텍스트 형식의 스크립트 파일입니다. 하지만 이 문서에서는 혼동될만한 다른 진짜 ‘실행’ 파일을 언급하지 않기 때문에 편의상 이 파일을 실행 파일이라고 부릅니다.

계좌[변경] 가계부 관리 프로그램 내에서, 회원이 직접 생성한 혹은 개인용으로 미리 주어진, 가계부 관리 기능이 탑재된 계좌. 세간에 널리 알려져 있는 ‘계좌’와는 달리 가계부 관리(태그 사용, 이전 내역 수정 등)를 수행할 수 있다.

데이터 파일 (혹은 파일)[특정] 가계부 관리 프로그램이 자기 자신의 여러 기능을 정상적으로 구동하기 위해 읽어들이는 텍스트 형식의 모든 파일들을 지칭하는 말. 가계부 관리 프로그램 속에 내재되어 있는 기능에 따라 두 가지의 데이터 파일이 존재합니다. 그러므로 이 문서에서 ‘데이터 파일’이라는 용어는 이러한 두 가지 데이터 파일 모두를 지칭합니다.

회원[명확] 회원가입 절차에서 입력한 회원 정보(이름과 주민등록번호)와 계정(ID와 비밀번호)의 주인을 특정하는 말. 여기서 주인은 자신이 만든 계정을 통해 프로그램에 로그인할 수 있고, 주인의 회원 정보는 따로 저장된다. 이 문서에서 ‘사용자’는 실제로 프로그램과 동적으로 상호 작용하는 ‘사람’을 지칭하며, ‘회원’은 어떤 생성된 어떤 계정과 그에 해당하는 회원 정보의 주인을 지칭하는 정적의 용어입니다.

회원 관리 파일[신규] 모든 회원들의 각 회원 정보와 계정, 해당 회원의 계좌 목록(계좌 번호)를 기록한 텍스트 형식의 파일.

계좌 정보 파일[신규] 가계부 관리 프로그램 안에 있는 모든 계좌에 대해, 각 계좌를 사용할 수 있는 회원들의 이름과 권한 현황, 태그 종류와 수입/지출 내역을 기록한 텍스트 형식의 파일. 즉, 하나의 계좌당 하나의 계좌 정보 파일이 대응됩니다.

태그[특정] 가계부 관리 프로그램의 핵심이라고 할 수 있는 각각의 수입/지출 항목에 붙어서, 그 항목의 속성을 나타내는 데이터. 태그는 상위 태그와 하위 태그로 분류되는데, 상위 태그는 포괄적인 개념이고 하위 태그는 각 상위 태그의 의미 영역에 포함되어 구체화적인 의미를 특정하는 개념입니다. 즉, 하나의 상위 태그에는 그에 부합하는 하위 태그가 여러 개 있으며, 이러한 하위 태그들은 오직 자신과 부합하는 상위 태그와만 함께 쓰일 수 있습니다.

태그 목록신규 해당 계좌에 존재하는 모든 태그를 상/하위 계층으로 구분하여 보여주는 목록의 형태를 의미합니다.

태그 위치신규 출력된 태그 목록에서 각 태그에 매칭된 번호 표현을 의미합니다. 해당 표현은 상위 태그는 <양의 정수>이고, 하위 태그는 <상위태그>.<양의정수> 형식입니다.

숫자특정 (서)아라비아 숫자들 중 표준 키보드로 직접 입력할 수 있는 10개 (U+0030 '0' ~ U+0039 '9')만을 뜻합니다. 즉, 이 문서에서 언급되는 "숫자"에는 로마 숫자나 각 언어별 고유 숫자 기호들이 포함되지 않으며, 아라비아 숫자들 중에서도 전각 숫자, 원 · 괄호로 둘러싸인 숫자, 위 · 아래 첨자용 숫자, 수식용 글꼴별 숫자 등은 역시 모두 포함되지 않습니다.

표준 공백신규 (혹은 **스페이스**space명확) 표준 키보드의 Spacebar로 직접 입력할 수 있는 문자 (U+0020 ' ' Space). 이 문서에서 표준 공백을 시각적으로 명확히 구분해야할 때에는 그 자리에 '␣' 기호를[1] 써서 표시합니다. 물론 실제로 해당 기호의 입력은 표준 공백을 입력하는 경우와는 다르게 동작하니, 이 문서의 예시를 긁어서 프로그램이나 데이터 파일에 붙여넣을 때 주의하기 바랍니다. 단지 표준 공백을 시각화 하기 위한 목적으로만 해당 기호를 사용할 뿐입니다.

개행특정 텍스트 형식의 파일을 편집할 때 표준 키보드의 Enter키로 입력할 수 있는 문자 혹은 문자들 (U+000A LF, U+000D CR). 사실은 이들 둘 외에도(Enter 키로 입력할 수 없는) U+000B Vertical Tab이나 U+000C Form Feed 등 넓은 의미의 개행에 포함되는 문자들이 더 있지만, 이런 문자들을 혹시 언급할 일이 있으면 반드시 ("개행"이 아니라) 각각의 구체적인 이름으로 부르겠습니다. 문서나 화면에서 개행은 대체로 2차원적으로 표현되므로 시각적으로 혼동할 여지가 적지만, 간혹 문서나 화면 폭 한계에 의한 자동 줄바꿈line wrap과의 구분을 명확히 해야할 때에는 개행 직전 줄의 맨 끝에 '↵' 기호를 덧붙여 표시합니다.

탭특정 (혹은 **가로 탭**기존) 텍스트 형식의 파일을 편집할 때나 IDLE, 일부 터미널 등에서 표준 키보드의 Tab↹ 키로 입력할 수 있는 문자 (U+0009 Character Tabulation). 사실은 세로 탭(U+000B)도 (넓은 의미의 개행에 포함되면서, 동시에) 탭의 일종이긴 하지만, 만일 언급할 일이 있으면 반드시 "세로 탭"이라고만 부르겠습니다. 이 기획서에서 탭을 시각적으로 명확히 표시해야할 때에는 '↹' 기호[2]로 표시합니다.

---

[1] 이 기호는 표준 공백을 시각적으로 드러내는 의미로 (특히 컴퓨터 분야에서는) 널리 통용되는 기호지만, 이 기호 '문자' 자체는 표준 공백 문자가 아니라 별개의 다른 문자 (U+2423 '␣' Open Box)입니다. 따라서, 실제 프로그램의 입력 프롬프트나 데이터 파일에 표준 공백 대신 이 기호를 입력하면 진짜 표준 공백을 입력하는 경우와는 다르게 동작하니, 이 문서의 예시를 긁어서 프로그램이나 데이터 파일에 붙여넣을 때 주의하기 바랍니다.

[2] 이 기호는 탭을 시각적으로 드러내기 위해 이 문서에서만 특별히 사용히 되는 것입니다. 해당 문자는 실제 탭과는 전혀 상관이 없는 별개의 다른 문자(U+21E5 '↹' Rightwards Arrow to Bar)입니다. 따라서, 실제 프로그램의 입력 프롬프트나 데이터 파일에 탭 대신 이 기호를 입력하면 진짜 탭을 입력하는 경우와는 다르게 동작하니, 이 문서의 예시를 긁어서 프로그램이나 데이터 파일에 붙여넣을 때 주의하기 바랍니다.

공백 (혹은 **whitespace**)명확 Python 문자열의 .isspace() 메서드가 참을 반환하는 문자들만 공백이라고 부르는 것으로 엄밀하게 정합니다.

(문자열의) 길이명확 이 문서에서 '문자'란 유니코드 문자만을 의미하고, '문자열'이란 유니코드 문자들의 나열을 의미합니다. 따라서, 문자열의 길이는 그 속에 들어있는 유니코드 문자들의 실제 개수로만 셉니다. 각 글자들의 메모리 상의 용량(즉, 바이트 수)이나 화면 상의 시각적인 폭 정보(W, FW, N, HW, A, Neutral)와는 전혀 무관합니다. 예를 들어, "안녕하세요"와 "Hello"의 길이는 똑같이 5입니다.

공백열^0 (혹은 **공백열**)신규 오직 공백whitespace들로만 구성된 길이 0이상의 문자열

공백열^{1}신규 오직 공백whitespace들로만 구성된 길이 1이상의 문자열

비개행공백열^0 (혹은 **비개행공백열**)신규 오직 '개행 이외의 공백'들로만 구성된 길이 0 이상의 문자열

비개행공백열^{1}신규 오직 '개행 이외의 공백'들로만 구성된 길이 1 이상의 문자열

실상 문자신규 (實像文字) 공백whitespace이 아니면서 표시가능printable한[3] 문자

단어변경 오직 실상 문자들로만 구성된 길이 1 이상의 문자열. 즉, 단어 속에는 공백이 나 표시불가능한 문자들이 전혀 들어있지 않습니다.

단어열^{1}신규 1개 이상의 단어들이 비개행공백열^{1}을 경계로 나열되어있는 문자열

명령어특정 (혹은 **명령**특정) 사용자가 7절의 주 프롬프트에 입력한 문자열 속에서 단어 형태로 특정한 위치에 (주로 첫번째, 간혹 두번째 단어로) 등장하면, 프로그램이 이를 특정한 의미로 간주하도록 예약된 총 28가지 단어들. 구체적인 목록은 7절의 표1에 나옵니다.

(명령어의) 동의어변경 어떤 명령어와 서로 완전히 같은 의미로 취급되는 다른 명령어.

동의어군 (혹은 **명령어군**)신규 동의어들끼리만 빠짐없이 모두 모은 묶음. 프로그램에는 총 7종류의 동의어군이 있으며[4] 구체적인 분류는 역시 7절의 표1에 나옵니다.

표준 명령어 (혹은 **대표 명령어**)신규 다른 명령어의 앞쪽 진부분문자열도[5] 아닌 명령어들. 각 명령어군마다 2개씩 (숫자 1개 + 로마자 단어 1개) 총 14개가 있으며, 구체적인 목록은 7절의 표1의 붉은색 명령어들입니다.

---

[3] 공백이 아닌데도 여전히 표시불가능한 문자들 중에는 U+0007 Alert (BEL) 같은 문자들이 있습니다.
[4] 결국 서로 다른 의미로 간주되는 명령어는 총 7종류라는 뜻입니다.
[5] '앞쪽 부분문자열'이라고 쓰지 않고 '앞쪽 진부분문자열'이라고 명시했음에 주의하기 바랍니다. 예를 들어, "find"는 비록 "find" 의 앞쪽 진부분문자열은 아니지만, 여전히 "find"의 앞쪽 부분문자열이기는 합니다. (모든 문자열은 자기 자신의 부분문자열입니다.)

인자[특정] 사용자가 7절의 주 프롬프트에 입력한 문자열 중 문법 형식 첫번째 항목의 ⟨단어열[1]⟩ 속에 들어있는 각 단어들. 다시 말해서, 첫번째 명령어 뒤에 비개행공백열[1]을 경계로 나열한 단어들이며, 프로그램은 첫번째 명령어가 같더라도 그 뒤의 인자들이 무엇인지에 따라 다르게 동작할 수 있습니다.

3. 기본 사항

3.1 작동 환경

기본적으로 Python 3 표준 인터프리터가 설치된 MS Windows 10의 cmd 창, PowerShell 창 및 Python IDLE 창에서 확실히 정상 작동합니다. 그 외에도, 아래 세 가지 요건들을 모두 만족시키면 정상적으로 작동할 수 있습니다 :

- Python 3 표준 인터프리터 (CPython)가 필요합니다. Python 3 비표준 인터프리터 (PyPy3등)로도 어쩌면 작동할 수도 있지만 보장되지는 않으며, Python 2 인터프리터로는 분명히 작동하지 않습니다.
- 사용자 계정별 홈 경로를 제공하고 환경변수를 통해 이를 특정할 수 있는 운영체제가 필요합니다. Linux와 MS Windows 10은 분명히 이 요건을 충족시키고, MS Windows Vista, 7, 8과 Apple OS X, macOS도 어쩌면 이 요건을 충족시킬 수도 있지만 보장되지는 않습니다.
- 유니코드 문자집합 입출력이 가능한 문자 기반 터미널이 필요합니다. 단, 터미널 인코딩이 꼭 UTF-8 이어야만 할 필요는 없으며, 어떤 인코딩으로 변환하든 간에 유니코드 문자로 인식할 수만 있으면 됩니다.
    - PuTTY, ConEmu, Gnome Terminal, Konsole 등 대부분의 최신 터미널 에뮬레이터들은 이 요건을 충족시킵니다.
    - gVim 창 내부에서 ':terminal'로 띄운 cmd 셸이나 Zsh 프롬프트에서도 (tenc 설정이 아예 부적 절하지만 않다면) 가계부 관리 프로그램을 정상적으로 작동시킬 수 있습니다. 어쩌면 Emacs GUI 창 내부에서 'M-x term'으로 띄운 셸 프롬프트에서도 작동될 수도 있지만, 보장되지는 않습니다.

3.2 프로그램 구성

- 프로그램이 배포될 때에는 주 실행 파일인 AccountBook.py 파일 하나만 배포됩니다.
- 프로그램이 올바르게 실행되면, 홈 경로에 "account-data"라는 이름의 데이터 파일(폴더)이 있는지 확인하고 없으면 생성합니다. 즉, 데이터 파일의 전체 경로는 "{HOME}₩account-data" 입니다. 이 파일은 사용자가 직접 수동으로 지우기 전까지는 (즉, 프로그램을 통해서는) 지워지지 않습니다.
- 회원 관리 파일과 계좌 정보 파일은 account-data 파일 안에 저장됩니다. 즉, 프로그램에서 저장하는 모든 파일은 해당 폴더 안에 위치합니다.

3.3 프로그램 설치 및 실행

기본적으로, 운영체제와 상관 없이 :

- 아무 경로에나 프로그램 주 실행 파일을 복사하여 설치합니다.
- IDLE Editor 창으로 주 실행 파일을 열고, F5 키를 누르거나 메뉴에서 Run → Run Module 항목을 누르면 IDLE Shell 창에서 실행됩니다.
- 혹은, 탐색기나 노틸러스 등 GUI 파일 관리창에서 (*.py 파일에 대한 연결 프로그램 설정이 적절하다면) 주 실행 파일을 더블클릭하여 실행할 수도 있습니다.

MS Windows의 경우, 추가로 :

cmd 창, PowerShell 창, 혹은 그에 준하는 터미널에서 주 실행 파일의 절대 경로나 상대 경로를 입력하여 실행할 수 있습니다.이때 확장자 (.py)는 생략할 수도 있습니다.

```
PS D:₩work₩SoPrj₩AccountBook> .₩AccountBook[.py] ↲
```

만일 %PATH% 환경변수에 들어있는 경로에 주 실행 파일을 설치했다면, 그 설치 경로나 현재 경로에 상관 없이 어디서나 주 실행 파일의 파일명만 입력하여 실행할 수 있습니다.

```
PS D:₩> AccountBook [.py] ↲
```

Linux의 경우 (와 어쩌면 macOS도), 추가로 :

대부분의 터미널에 띄운 대부분의 셸 프롬프트에서, python 명령 뒤에 주 실행 파일의 절대 경로나 상대 경로를 입력하여 실행할 수 있습니다.

```
reeseo@iserlohn:~/work/SoPrj/AccountBook$ python AccountBook.py ↲
```

만일 $PATH 환경변수에 들어있는 경로에 주 실행 파일을 설치했고 주 실행 파일에 자신의 실행 (x) 권한을 설정했다면, 그 설치 경로나 현재 경로에 상관 없이 어디서나 주 실행 파일의 파일명만 입력하여 실행할 수 있습니다.

```
reeseo@iserlohn:~$ AccountBook.py ↲
```

4. 프로그램 사용 흐름도

프로그램 사용 흐름, 즉, 프로그램 사용자 관점에서 프로그램을 사용하는 단계별 경로는 다음과 같습니다.

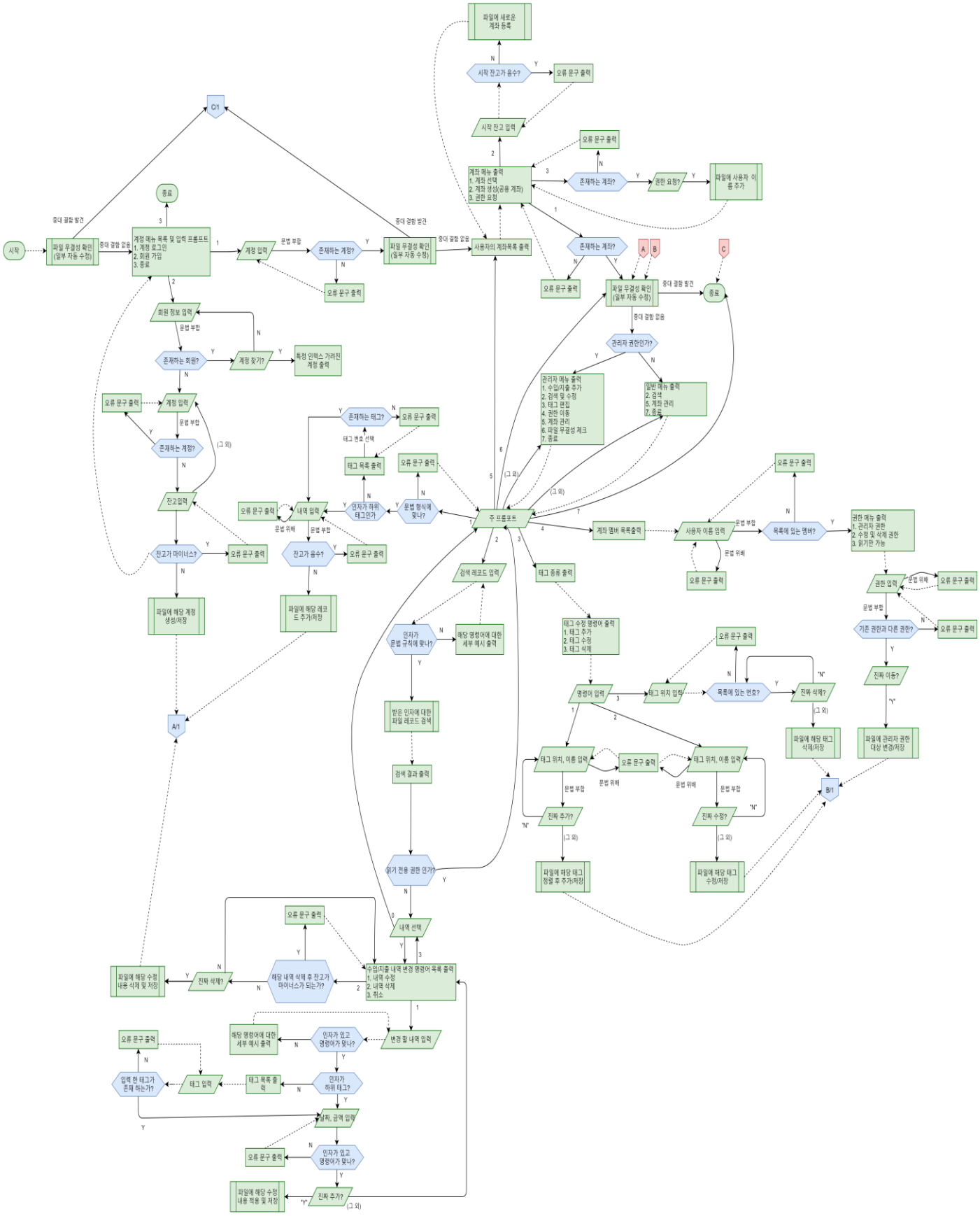

그림1: 사용 흐름도

5. 데이터 요소

데이터 요소에는 크게 날짜, 금액, 태그, 회원 정보, 계정 정보가 있습니다.

각 요소마다 동치비교 규칙도 서로 다르고 검색시 매칭 규칙도 서로 다르므로, 아래의 각 요소별 소절에서 해당 요소의 동치 비교 규칙, 검색시 매칭 규칙을 각각 명시하겠습니다. 단, 정렬시의 순서 비교 방법은 모든 데이터 요소들이 공통적으로 '파일에 저장된 문자열 그대로, 등록 순서로' 비교됩니다.

5.1 날짜

문법 규칙 : 문법적으로 올바른 날짜는 다음 형식들 중 하나에 부합하는 문자열입니다.

- 숫자 2개 또는 4개 + '-' 문자 + 숫자 2 개 + '-' 문자 + 숫자 2 개 (예: "2021-03-10")
- 숫자 2개 또는 4개 + '/' 문자 + 숫자 2 개 + '/' 문자 + 숫자 2 개 (예: "2021/03/10")
- 숫자 2개 또는 4개 + '.' 문자 + 숫자 2 개 + '.' 문자 + 숫자 2 개 (예: "2021.03.10")
- 위 1 ~ 4 항 중 하나 속의 모든 '-', '/', '.' 문자들을 제거한 문자열 (예: "210310" 혹은 "20210310")

이 때, 구분자와 숫자 외의 어떤 문자(공백 포함)도 사용할 수 없고, 연도는 4자리, 월, 일은 2자리로 적는 것만을 허용합니다. 즉, 다음 예시는 문법 규칙에 맞지 않는 경우로 취급합니다.

```
날짜: 21.1.2↵
날짜: 21␣01␣02↵
```

해석 : 문법 규칙을 통과한 날짜는 다음과 같이 해석합니다.

- 날짜 속에 구분자 ('-', '/', '.')가 있을 경우, 모든 구분자를 지웁니다.
- 날짜 속에 구분자 ('-', '/', '.')가 없는 상태에서, 맨 앞 네 숫자를 '연도'로, 그 다음 두 숫자를 '월'로, 그 다음 두 숫자를 '일'로 간주합니다.

의미 규칙 : 의미상 올바른 날짜는, 해석된 연·월·일을 기준으로 연도가 1970년 이후(UNIX Epoch 시각)부터 2037년 이전(32비트 시스템) 까지여야 하며, 해당 날짜가 현행 그레고리력에 존재하는 날짜여야 합니다.

검색 : 검색 명령의 인자를 검색어로 파일에 저장된 날짜 중에서 찾을 때에는, 위와 동일한 문법 규칙, 해석과 의미 규칙을 거쳐 완전히 동일한 경우에만 합치된 것으로 간주하되, 예외적으로 4자리로 '연도'만 입력하거나, 6자리로 '연도'와 '월'만 입력하는 것을 허용합니다. 즉, 검색 시 날짜는 4자리, 6자리, 8자리 중 하나로 작성이 가능합니다. 4자리로 '연도'만 입력한 경우 해당 '연도'의 모든 내역들과 합치된 것으로,

6자리로 '연도'와 '월'만 입력한 경우 해당 '월'의 모든 내역들과 합치된 것으로 간주합니다. '연도'를 생략하거나, '연도'와 '월'을 생략하고 '일'만 작성하는 경우는 허용되지 않습니다. 문법 형식에서 연·월·일의 자릿수를 정해두고, 앞에서부터 연도, 월, 일 순으로 해석한 이유는 이를 위함이었습니다. 예를 들어, "201212"는 2020년 12월 12일이 아닌 2012년 12월로 해석하며, 이는 2012년 12월의 모든 내역들과 합치됩니다.

검색 시 두 개의 날짜를 입력하는 경우, 둘 사이의 모든 날짜들에 대해서도 합치되는 것으로 간주합니다. 이 때, 입력하는 두 개의 날짜의 순서는 무관합니다. 예를 들어, "2021.01.15 2021.01.012021.01.01 2021.01.15"는 2021년 1월 1일부터 15일까지의 모든 내역에 합치됩니다. "2021.01 2021.04"는 2021.01이 2021년 1월 1일부터 31일까지의 내역과 합치되고, 2021.04가 2021년 4월 1일부터 4월 30일까지의 내역과 합치되므로, 그 사이인 2월 1일부터 3월 31일까지의 내역과도 합치됩니다. 즉, 2021년 1월 1일부터 2021년 4월 30일까지의 모든 내역과 합치됩니다.

검색 시 인자가 없을 경우, 작성시점을 기준으로 최근 10개의 내역의 날짜들과 합치되는것으로 간주합니다. 총 내역이 10개 이하일 경우 모든 내역과 합치되는 것으로 간주됩니다.

5.2 금액

금액은 사용자가 공용가계부 혹은 개인가계부에 입력할 수익내용 혹은 지출내용 혹은 남은 잔고를 표현합니다.

문법 규칙 : 문법적으로 올바른 금액 입력 형식은 아래 조건 모두에 부합하는 문자열입니다.

- 길이가 1이상이며 1개 이상의 숫자와 ‘,’ , ’-’ , ’+’ 그리고 ‘원’으로만 구성 되어야함 [1]
- 문자열 사이(문자열의 처음과 끝을 제외)에는 ‘,’를 제외한 어떠한 문자도 허용하지 않음.
- ‘-’, ‘+’는 문자의 맨 앞에만 올 수 있음.
- 문자열의 길이가 1보다 클 때 마지막 문자는 숫자 혹은 원만 허용함.
- 문자열의 길이가 1이라면 그 문자는 무조건 숫자여야 함.
- 문자열의 첫 시작은 숫자 혹은 ‘-’ 혹은 ‘+’여야 함.

[1] Python 특성상 문자열 길이의 실질적인 최대 한계는 실행 환경과 상황 등에 따라 (비록 이론적인 최대치는 있지만, 그보다 짧게) 달라지므로, 이 문서에서 규정하지 않습니다. 이후 ‘문자열의 길이를 특별히 제한하고 있지 않은 문법 규칙에 대해서도 마찬가지입니다.

예를 들어, “01000원”이나 “100000,000”, “+01000원” 심지어 “1” 는 모두 올바른 금액 입력 방식이지만, “1000원.1000”이나 “1000$”,“₩1000”, ”-”는 올바른 금액 입력 형식이 아닙니다.

의미 규칙 : 입력하려는 금액이 지출내역이라면(금액 문자열의 첫시작이 ‘-’ 라면) 입력한 금액은 사용자의 잔고에 있는 금액보다 커서는 안됩니다. 즉, 지출 내역 추가 시 입력한 금액은 입력한 금액의 지출 시점과 가장 최근에 계산되어 있는 내역 시점 사이에 지출내역에 쓰여있는 잔고들을 넘으면 안됩니다. 또한, 지출내역 수정시에는 (원래 쓰여있던 지출금액) - (수정하려는 지출 금액)이 입력한 금액의 지출 시점과 가장 최근에 계산되어 있는 내역 사이에 있는 지출내역에 쓰여있는 잔고들을 넘어서는 안됩니다.

예를 들어, (날짜, 지출금액, 태그, 상세내용, 잔고 순으로)

(A) 2021-01-05 -1000 [음식][간식] 콘칩 2000
(B) 2021-02-11 -100000 [선물][데이트] 반지 20000
(C) 2021-03-09 -8000 [음식][해장] 순대국밥 12000

이렇게 3가지의 데이터가 저장되어있을 때, A의 지출내역을 갑자기 -100000으로 바꾸는 경우, B와 C는 잔고에 없는 돈을 쓴 경우이므로 의미상 오류입니다.

5.3 태그

사용자는 프로그램에 기본적으로 내장되어 있는(Built-in) 태그들 이외에 다른 태그의 사용을 충분히 원할 수 있습니다. 그러므로 해당 프로그램에서는 사용자가 직접 태그를 추가할 수 있게 합니다. 아래는 사용자가 추가하려는 태그의 이름에 대한 키 입력 규칙입니다.

문법 규칙 : 다음 조건들에 모두 부합하는 문자열만 태그 이름의 문법 규칙에 맞는 경우로 취급합니다.

- 시작은 ‘[’ 문자이며, 끝은 ‘]’ 문자여야만 함,하며, 문자열 사이에 ‘]’ 문자가 포함될 수 없음
- 탭, 개행의 포함을 허용하지 않음.

즉, 다음 예시와 [새싹␣반⇥간식] 같은 문자열은 문법 규칙에 맞지 않는 경우로 취급합니다.

위와 같이 사용자가 태그 이름 문법 규칙에 맞지 않는 문자열을 입력했을 경우에는 오류 메시지를 출력합니다. 이에 대한 예시는 7.5 태그 편집 명령어군에서 볼 수 있습니다.

해석 : 문법 규칙을 통과한 문자열에 다음의 처리를 차례대로 가합니다.

1. [<비개행공백열[1]><문자열><비개행공백열[1]>]형식이면, <비개행공백열[1]>을 모두 지움.
2. 남은 문자열에 2개 이상의 연속된 공백이 있다면 1개의 공백으로 수정.

예를 들어, 문법 규칙을 통과하고 넘어온 문자열이 “[분␣␣식]”이라면, 이는 “[분␣식]”으로 해석합니다. 해당 경우는 <문자열> 양쪽에 <비개행공백열[1]>이 모두 없는 경우였습니다. 만약 “[␣분␣식]”이었다면, “[분␣식]”으로 해석됩니다. “분␣식”이 <문자열>에 해당하고, <문자열> 왼쪽에 있는 ␣이 <비개행공백열[1]>인 것입니다(여기서의 모든 “, ” 문자는 문자열임을 표시하기 위함입니다).

지금부터 해석을 거친 후 얻은 문자열을 “수정된 문자열”이라고 정의합니다. <비개행공백[1]>을 제거하는 이유는, 애초에 그런 형태를 틀리다고 간주하기 때문입니다. 그런데도 프로그램은 <비개행공백열[1]>을 모두 제거해줌으로써 사용자에게 일종의 배려를 해주는 것입니다.

의미 규칙 : 다음 조건을 만족하는 수정된 문자열만 태그 이름의 의미 규칙에 맞다고 봅니다.

- 길이가 20이하여야 함.
- 기존 태그들의 이름과 중복되지 않아야 함.
- 숫자나 특수문자의 조합으로만 구성되어 있지 않아야 함.

예시로 다음의 수정된 문자열들은 모두 의미 규칙에 맞는 경우로 취급합니다.

```
[광고␣선전비] (수정된 문자열 길이 : 8) ↵
[법인세비용차감전순이익] (수정된 문자열 길이 : 13) ↵
[종업원#1␣급여] (수정된 문자열 길이 : 10) ↵
[박철␣헤어␣커커@] (수정된 문자열 길이 : 11) ↵
[영업␣3부] (수정된 문자열 길이 : 7) ↵
```

물론 위 예시의 모든 수정된 문자열은 기존의 태그 이름들과 중복되지 않음을 전제로 합니다. 또한 태그의 시작과 끝 문자인 특수 문자 또한 문자열의 길이에 포함됩니다. 또한 아래의 예시는 의미 규칙에 맞지 않는 수정된 문자열들을 보여줍니다.

```
[5000] (숫자로만 구성) ↵
[***%***] (특수문자로만 구성) ↵
[23^23] (숫자와 특수문자로만 구성) ↵
```

이때 의미 규칙에 위반되는 수정된 문자열은 1, 2번을 동시에 어겼거나, 1번만 어겼거나, 2번만 어겼거나, 3번만 어겼을 수 있습니다. 이러한 네 가지 이외 상황에 해당하는 틀린 수정된 문자열의 사례는 존재할 수 없습니다. 모든 기존 태그들(built-in 태그 포함)은 태그 이름 의미 규칙의 1, 2번을 이미 모두 만족하고 있습니다. 그러므로 의미 규칙 3번에 위배된 수정된 문자열은 의미 규칙 1번 또는 2번과 동시에 위배될 수 없습니다. 의미 규칙에 맞지 않은 수정된 문자열을 얻었을 경우, 해당 수정된 문자열이

어긴 의미 규칙에 따라 오류 메시지를 출력합니다. 이는 마찬가지로 7.5 태그 편집 명령어군에서 확인할 수 있습니다.

동치 비교 : 태그의 동치 비교는 사용자가 새로운 태그를 추가할 때나, 검색 시 태그를 인자로 사용하여 내역을 찾을 경우 이루어집니다. 두 태그가 동치 비교될 시, 두 태그 이름의 문자열이 완벽히 동일한 경우에만 동치라고 판단합니다.

검색 : 검색 명령의 인자를 검색어로 파일에 저장된 내역의 태그 중에서 찾을 때는, 태그가 기존 태그 목록의 태그와 완전히 동치인 내역만 합치인 것으로 간주하되, 검색어가 상위 태그와 동치되는 경우 해당 상위 태그의 모든 하위 태그들을 사용한 내역들과도 합치되는 것으로 간주합니다.

5.4 회원 가입 절차에서

'회원 가입' 절차를 통해서 새로운 계정을 생성할 수 있고, 혹은 까먹은 계정에 대한 힌트 문자열을 얻을 수 있습니다. 일단 회원 가입 메뉴를 선택했다면, 회원 정보를 입력해야 합니다. 회원 정보는 '이름 + 주민등록번호' 형식인데, 이때 입력받은 회원 정보를 프로그램이 기존에 저장하고 있지 않았다면 신규 회원이 접근한 것입니다. 신규 회원은 계정을 생성해야 하므로 계정 입력 단계로 넘어갑니다. 만약 프로그램이 기존에 저장하고 있던 회원 정보라면, 해당 회원은 이미 계정을 만든 상태라는 것입니다. 계정이 있는데도 불구하고 로그인을 하지 않고 회원 가입을 선택했다는 것을 "계정을 까먹었기 때문"이라고 간주합니다. 그러므로 이 경우는 계정 찾기 단계로 넘어갑니다. 이러한 회원 가입의 흐름에서 적용되는 키 입력 규칙이 두 가지 있습니다. 바로 회원 정보와 계정 입력에 대한 키 입력 규칙입니다. 그런데, 회원 정보 키 입력 규칙을 정의하기 위해선 먼저 "이름"과 "주민등록번호"를 정의해야 합니다.

5.4.1 회원 정보(이름, 주민등록번호)

1) 이름

문법 규칙 : 문법적으로 올바른 이름은 아래 세 조건을 모두 만족시키는 문자열입니다.

- 길이가 1 이상
- 첫 문자와 마지막 문자는 실상 문자
- 탭도 개행도 전혀 들어있지 않음

예를 들어, (따옴표 없이) "홍␣길동"과 "Hong gil dong", "H", 심지어 "!@#$"도 모두 문법적으로 올바른 이름입니다.

의미 규칙 : 이름에는 아무런 추가적인 의미 규칙도 없습니다.

동치 비교 : 두 이름은 회원 정보를 비교할 때 동치 비교됩니다. 이때 비교되는 두 이름 문자열이 완벽히 동일한 경우(로마자 대소문자와 공백까지 모두)에만 동치라고 판단합니다.

2) 주민등록번호

문법 규칙 : 문법적으로 올바른 주민등록번호는 아래 형식에 부합하는 문자열입니다.

■ **<6자리 생년월일 숫자><표준 공백><{1,2,3,4} 중 1개의 숫자><6자리 숫자>**

화살괄호<, >나 중괄호 {, }는 실제 문자가 아니라 문법 설명용 표식입니다. 6자리 생년월일 숫자란, 예를들어 2003년 2월 7일이나 1903년 2월 7일 모두 030207로 표기합니다. 이는 주민등록번호 앞 6자리에 해당하는 부분입니다. 그 다음에는 표준 공백이 옵니다. 이는 주민등록번호의 앞 6자리와 뒤 7자리를 구분하는 구분자의 역할을 합니다. 표준 공백 다음에는 1, 2, 3, 4 중 하나의 수만 들어올 수 있습니다. 1은 1900년대 태어난 남자, 2는 1900년대 태어난 여자, 3은 2000년대 태어난 남자, 4는 2000년대 태어난 여자를 의미합니다. 이후에는 정확히 6자리 숫자만 와야합니다. 해당 6자리 숫자는 아무런 수들로 이루어져 있어도 상관이 없습니다. 이처럼 주민등록번호는 오직 숫자와 표준 공백으로만 구성되는 문자열입니다.

의미 규칙 : 문법 규칙을 통과한 주민등록 번호에 대해, 다른 사용자들의 주민등록번호와 중복되지 않는 경우에만 의미 규칙에 맞습니다.

동치 비교 : 두 주민등록번호는 회원 정보를 비교할 때 동치 비교됩니다. 이때 비교되는 두 주민등록번호 문자열이 완벽히 동일한 경우에만 동치라고 판단합니다.

이름과 주민등록번호를 정의했으므로, 이제는 회원 정보의 키 입력 규칙을 설명하겠습니다. 회원 정보란 바로 아래의 형식을 만족하는 문자열을 말합니다.

■ **<이름><비개행공백열[1]><주민등록번호>**

화살괄호<, >는 실제 문자가 아니라 문법 설명용 표식입니다. 즉, 회원 정보는 이름의 첫 문자(실상 문자)로 시작하고, 주민등록번호의 마지막 숫자로 끝나야 합니다. 이름과 주민등록번호 사이에는 그 둘을 구분짓는 구분자로서 비개행공백열[1]이 존재합니다. 즉, 사용자가 회원 정보로서 입력한 문자열은 비개행공백열[1]을 기준으로 앞 부분은 이름으로, 뒷부분은 주민등록번호로 확실히 판정되어야 합니다. 이를 만족하지 못한다면 회원 정보로 받아들일 수 없습니다. 올바른 회원 정보들간에서는 동명이인이라고 할지라도 주민등록번호로 어떤 개인을 완벽히 특정할 수 있게 됩니다. 두 회원 정보 간의 동치 비교를 할 때에는, 두 회원 정보의 이름과 주민등록번호가 모두 동치여야지만 두 회원 정보도 동치라고 판단합니다. 예를 들어, 회원 정보1의 이름은 “홍길동”이고, 주민등록번호는 “031121␣3029481”입니다. 회원 정보2의 이름은 “홍길동”이고, 주민등록번호는 “040503␣3092817”입니다. 두 회원 정보는 이름은 동치이지만 주민등록번호는 동치가 아니므로, 서로 다른 회원 정보입니다.

사용자가 회원 정보를 입력했을 경우, 프로그램은 회원 정보의 동치 비교를 통해 그 회원 정보가 이미 저장된 데이터인지를 확인합니다. 만약 회원인 경우(이미 저장된 경우)는 이전에 말했듯이 계정 찾기 단계로 넘어가고, 회원이 아닌 경우는 새로 생성할 계정을 입력받습니다. 계정 찾기 단계는 따로 키 입력을

받지 않습니다. 그저 입력받은 회원 정보에 해당하는 계정을 특정 인덱스(ID, 비밀번호 모두 뒤 3자리)만 '*' 문자로 대체하여 출력해줍니다. 이는 까먹은 해당 계정에 대한 힌트라고 볼 수 있습니다.

```
회원 정보를 입력해주세요 : 홍길동␣011029␣3019583↵
이미 회원이십니다. 계정 찾기를 원하십니까?(y/s) : y↵
ID : cetap***↵
비밀번호 : Cof#12***↵
```

이처럼 ID와 비밀번호의 뒷 3자리를 '*' 문자로 대체하여 출력합니다. 이를 보고 이에 해당하는 계정이 무엇이었는지에 대한 힌트를 얻을 수 있습니다.

5.4.2 계정 정보(ID, 비밀번호)

1) ID

문법 규칙 : 다음 조건들에 모두 부합하는 경우만 문법 규칙에 맞다고 취급합니다.

- 문자열의 길이가 5이상 20 이하여야 함.
- 영문 소문자 및 숫자와 특수기호(_),(-)로만 구성되어야 함.
- 첫 문자는 특수기호가 아니어야 함.

예시로 몇 개의 문자열에 대한 ID 문법 규칙 적용 결과를 아래에 제시합니다.

```
ID : kerastyle (문법 규칙에 맞음) ↵
ID : happy_2-_-- (문법 규칙에 맞음) ↵
ID : 2_--_538 (문법 규칙에 맞음) ↵
```

```
ID : Pythonic_code1 (문법 규칙에 맞지 않음[영문 대문자 사용]) ↵
ID : -coldfeet24 (문법 규칙에 맞지 않음[첫 문자가 특수기호임]) ↵
ID : 4--_ (문법 규칙에 맞지 않음[문자열의 길이가 5 미만임]) ↵
ID : mineral23@ (문법 규칙에 맞지 않음[불허된 특수기호 '@' 사용]) ↵
```

ID 문법 규칙에 위배되는 문자열에 대해서, 해당 문자열이 어긴 문법 규칙 조항을 모두 알리는 메시지를 출력합니다.

```
ID : Math↵
사용할 수 없는 ID입니다. 아래의 사항을 확인해주세요.
1. 5~20자리로 입력해주세요.
2. 영문 소문자 및 숫자, 특수 기호(_),(-)로만 구성해주세요.
```

ID : -cocoball↵
사용할 수 없는 ID입니다. 첫 문자는 특수 기호가 아니어야 합니다.

의미 규칙 : 문법 규칙을 통과한 ID에 대해, 다음 조건들에 모두 부합하는 경우만 의미 규칙에 맞습니다.

- 다른 사용자들의 ID와 중복되지 않아야 함.

의미 규칙에 위반되는(즉, 다른 ID와 중복되는) ID에 대해서는 다음 메시지를 출력합니다.

ID : a123456↵
이미 사용중인 ID입니다.

동치 비교 : 두 ID를 동치 비교를 할 때에는, 두 ID의 문자열 전체가 (로마자 대소문자까지 포함해서) 서로 완전히 일치해야만 동치인 ID로 판단합니다.

2) 비밀번호

문법 규칙 : 다음 조건들에 모두 부합하는 경우만 문법 규칙에 맞다고 취급합니다.

- 문자열의 길이가 8이상 20 이하여야 함.
- 영문 대문자, 소문자, 숫자, 특수문자 중 3종류 이상의 조합으로 구성되어 있어야 함.

예시로 몇 개의 문자열에 대한 비밀번호 문법 규칙 아래에 제시합니다.

비밀번호 : SoftTrabel@ (문법 규칙에 맞음 : 영문 대문자/소문자/특수문자) ↵
비밀번호 : seq2seq2 (문법 규칙에 맞지 않음 [영문 소문자/숫자의 2종류 조합임]) ↵
비밀번호 : VAIV!482 (문법 규칙에 맞음 : 영문 대문자/소문자/숫자) ↵
비밀번호 : Ku92! (문법 규칙에 맞지 않음 [문자열 길이가 8미만임]) ↵

비밀번호 문법 규칙에 위배되는 문자열에 대해서, 해당 문자열이 어긴 문법 규칙을 모두 알리는 메시지를 출력합니다.

비밀번호 : love@↵
사용할 수 없는 비밀번호입니다. 아래의 사항을 확인해주세요.
1. 8~20자리로 입력해주세요.
2. 영문 대문자, 소문자, 숫자, 특수문자 중 3종류 이상의 조합으로 구성해주세요.

비밀번호 : a12@↵
사용할 수 없는 비밀번호입니다. 8~20자리로 입력해주세요.

의미 규칙 : 문법 규칙을 통과한 비밀번호에 대해, 다음 조건들에 모두 부합하는 경우만 의미 규칙에 맞습니다.

- Jaccard 거리로 ID와의 문자열 유사도를 비교한 결괏값이 0.4이하여야 함.

의미 규칙에 위반되는 비밀번호에 대해서는 다음 메시지를 출력합니다.

```
ID : a12345↵
비밀번호 : aa1234!!!!!↵
ID와 너무 비슷합니다.
```

Jaccard 거리는 두 집합 사이의 유사도를 측정하는 방법입니다. 보편적인 회원가입 절차에선 ID와 비슷하지 않은 비밀번호의 사용을 권장하는 경우가 많습니다. 본 프로그램에서도 사용자에게 그러한 사용을 권장하고자, ID와 비슷한 비밀번호의 사용을 불허합니다. 그러려면 ID와 비밀번호 문자열에 대한 유사도를 비교해야 하는데, 본 프로그램에선 그 방법으로 Jacaard 거리를 채택한 것입니다. 이때 비밀번호와 비교되는 ID는 이미 ID의 문법 규칙과 의미 규칙에 모두 맞다고 판정된 후입니다. 즉, 사용자는 먼저 적합한 ID를 정한 후에야 비밀번호를 설정할 수 있습니다.

동치 비교 : 두 비밀번호를 동치 비교를 할 때에는, 두 비밀번호의 문자열 전체가 (로마자 대소문자까지 포함해서) 서로 완전히 일치해야만 동치인 비밀번호로 판단합니다.

회원 가입을 마친 사용자로 프로그램에 접근하기 위해서는 로그인이 필요합니다. 로그인 과정에서 회원의 ID와 비밀번호를 입력받을 때 적용되는 키 입력 규칙은 계정 생성시 ID와 비밀번호의 키 입력 규칙과 완전히 동일합니다. 회원 가입이 완료된 회원 계정으로만 로그인할 수 있는데, 이를 위해 로그인의 ID 및 비밀번호 키 입력 규칙이 계정 생성시의 ID 및 비밀번호의 키 입력 규칙과 다를 이유는 전혀 없기 때문입니다(오히려 다르다면 이상할 것입니다).

6. 데이터 파일

기본적으로, 이 프로그램은 실행 중에 사용자로부터 회원 정보, 계정 정보, 잔고, 지출내역, 수입내역, 태그를 키 입력으로 받아들이고 이 정보들을 프로그램 스스로 내부 기능을 통해 데이터 파일에 저장합니다. 이런 방식으로 저장된 데이터파일은 사용자가 별도로 텍스트 파일을 이용하여 직접 생성/편집/저장하는 방식을 전혀 지원하지 않습니다. 즉, 프로그램에 관한 데이터 파일을 직접 생성/편집/저장할 경우, 정상 작동을 보장하지 않습니다.

6.1 의미 규칙

기본적으로 아래의의미 규칙을 만족하지 않는 파일은 올바른 데이터 파일이 아닙니다.

1. 계좌의 잔고는 양수를 유지해야 함 : 계좌의 잔고는 기본적으로 0보다 작은 수가 올 수 없고, 이 프로그램은 기본적으로 계좌의 잔고에 0보다 작은 수가 오는 경우 프로그램에 혹은 데이터 파일에 치명적 오류가 있다고 판단하고 프로그램을 종료 시킵니다. 허나 잔고 입력 부분만 잘못되어있는 경우를 고려하여 처음 내역부터 끝까지 계산해서 다시 잔고내역을 새로 고쳤을 때 잔고가 전부 0이상의 수라면 프로그램을 정상 작동합니다.

2. 사용자의 계좌는 최소 하나 이상 있어야 함 : 사용자의 계좌는 최초에 하나 자동 생성해주기 때문에 무조건적으로 하나 이상은 있어야 합니다. 만약 무결성 검사 시 없다면 잔고 0원의 계좌를 생성해주고, 생성에 실패하는 경우 프로그램을 즉시 종료합니다.

6.2 무결성 확인 및 처리

프로그램에는 두 개의 데이터 파일이 존재합니다. 바로 회원 관리 파일과 계좌 정보 파일입니다. 회원 관리 파일은 프로그램이 실행될 때 바로 무결성 확인 및 처리를 합니다. 계좌 정보 파일은 사용자가 각 계좌에 접근 (해당 계좌 선택) 했을 때 무결성 확인 및 처리를 합니다. 무결성 확인 및 처리에 대한 자세한 내용은 아래와 같습니다.

6.2.1 프로그램 실행될 때, 명령어가 입력되었을 때

첫번째로 프로그램이 실행될 때 혹은 주 프롬프트에 verify 명령어가 입력될 때 다음과 같이 데이터 무결성을 확인하고 오류가 확인될 시 치명적 오류를 알리고 프로그램이 종료됩니다 :

1. 사용자의 홈경로를 파악하려고 시도해서, 홈경로가 파악되면 아무출력없이 넘어가고, 파악되지 않으면 오류 메세지를 출력한 후 프로그램이 즉시 종료됩니다.

```
PS D:₩>AccountBook↵
!!! 오류: 홈경로를 파악할 수 없습니다! 프로그램을 종료합니다.
PS D:₩>
```

2. 홈 경로에 데이터 파일이 있는지 확인해서 :

   - 없으면 경고 문구를 출력하고, 홈 경로에 빈 데이터 파일 생성을 시도합니다. 이때 생성되는 Account-data파일은 “폴더”입니다. 만일 파일 생성에 성공하면 성공 메세지를 출력한 후 다음 단계로 넘어가고

```
PS D:₩> AccountBook ↵
..! 경고: 홈 경로 C:₩Users₩fourTeam₩ 에 데이터 파일이 없습니다.
... 홈 경로에 빈 데이터 파일을 새로 생성했습니다:
```

```
C:₩Users₩fourTeam₩Account-data
```

만일 파일 생성에 실패하면 오류메세지를 출력한 후 프로그램이 즉시 종료됩니다.

```
PS D:₩> AccountBook ↲
..! 경고: 홈 경로 C:₩Users₩fourTeam₩ 에 데이터 파일이 없습니다.
!!! 오류: 홈 경로에 데이터 파일을 생성하지 못했습니다! 프로그램을 종료합니다.
PS D:₩>
```

- 있으면 파일에 대한 입출력 권한이 있는지 확인합니다. 만일 권한이 있으면 아무 출력 없이 아래 단계로 넘어가고, 없으면 오류 메세지를 출력한 후 프로그램이 즉시 종료됩니다.

```
PS D:₩> AccountBook↲
!!! 오류: 데이터 파일
C:₩Users₩fourTeam₩ Account-data에 대한 입출력 권한이 없습니다! 프로그램을 종료합니다.
PS D:₩>
```

3. 오류 없이 이 단계까지 오면 첫번째 무결성 확인/처리가 완료된 것이고, 계정 메뉴 목록 및 입력 프롬프트를 출력하고 대기합니다.

6.2.2 계정 입력 후 존재하는 계정임을 확인한 후 무결성 확인 및 처리

두번째로 계정 입력 후 그 계정이 존재함을 확인한 후, 다음과 같이 데이터 무결성을 확인하고 오류가 있을 시 프로그램이 종료됩니다.:

1. 사용자의 계좌가 최소 하나 있는지 확인한 후 없다면 경고 문구를 출력한 후 잔고 0원의 계좌 생성을 시도합니다 :

   - 실패한다면 오류문구와 함께 프로그램을 즉시 종료합니다:

```
AccountBook>
ID : kerastyle↲
비밀번호 : VAIV!482↲
!!! 경고: 사용자의 계좌가 하나도 존재하지 않습니다.
!!! 오류:계좌 생성에 실패하였습니다. 프로그램을 종료합니다.
PS D:>
```

   - 성공한다면 다음 단계로 넘어갑니다 :

```
AccountBook>
ID : kerastyle↵
비밀번호 : VAIV!482↵
!!! 경고: 사용자의 계좌가 하나도 존재하지 않습니다.
... 사용자의 개인 계좌를 생성하였습니다:
AccountBook>
```

2. 이 단계까지 오면 두번째 무결성 확인/처리가 완료된 것이고, 사용자의 계좌목록을 출력해주고 계좌 메뉴를 출력해준 후 대기합니다.

6.2.3 계좌 선택 후 존재하는 계좌일 때 무결성 확인 및 처리

세번째로 계좌 선택 하고 해당계좌의 데이터 파일이 있음을 확인한 직후에, 다음과 같이 데이터 무결성을 확인하고 오류가 있을 시 프로그램이 종료됩니다 :

1. 데이터 파일에 해당하는 계좌 내역들 전부를 읽어서 잔고가 마이너스로 변한 곳이 있는지 확인해서 :

   ■ 해당 계좌의 내역을 처음부터 다시 잔고를 계산해 봐도 잔고가 0보다 작은 숫자인 데이터가 존재하면 프로그램이 즉시 종료됩니다.

```
AccountNumber > 계좌 선택: 1↵
..! 경고: 데이터 파일에 잔고에 이상이 있습니다!
!!! 오류: 데이터 파일에 이상이 생겨 프로그램을 종료합니다.
PS D:₩>
```

   ■ 해당 계좌의 내역을 처음부터 다시 잔고를 계산했을 때, 잔고가 정상적으로 전부 0이상의 숫자인 데이터라면

```
AccountNumber > 계좌 선택: 1↵
..! 경고: 데이터 파일에 잔고에 이상이 있습니다!
... 잔고를 새로 고침 하였습니다:
AccountNumber >
```

위 예시는 사용자가 이미 주 프롬프트에 위치한 상태에서, 계좌를 선택할 수 있는 명령을 내린 상황입니다. 즉, 이미 어떤 계좌에 접근한 상태에서, 이제는 그 계좌에서 다른 계좌로 접근하려는

상황입니다. 이러한 상황 말고도, 로그인 후 처음으로 계좌를 선택하는 상황에서도 해당 계좌에 대한 무결성 확인 및 처리를 당연히 진행합니다.

2. 오류없이 이 단계까지 오면 두번째 무결성 확인/처리가 완료된 것이고, 관리자 권한 유무에 따라 관리자 메뉴 혹은 일반 메뉴를 출력하고 대기합니다.

7. 주 프롬프트

주 프롬프트는 화면에 다음과 같이 출력되고, 사용자의 키 입력을 기다립니다 :

```
AccountNumber >
```

문법 규칙: 주 프롬프트에 키 입력하는 올바른 문법은 다음 두 형식 중 하나입니다:

- **<명령어>**
- **<명령어> <비개행공백열[1]> <단어열[1]>**

단, 위의 <명령어>자리에는 반드시 표 1의 28개 단어들 중 하나를 써야 하며, 이들만이 문법적으로 올바른 '명령어'입니다. 표 1에서 같은 칸 속에 들어있는 명령어들끼리는 모두 서로 동의어고, 각 칸은 명령어군 (동의어군)을 나타내며, 붉은색 명령어들은 다른 명령어의 축약형이 아닌 표준 명령어들이자 각 명령어군을 나타내는 대표 명령어들이기도 합니다.

1 add + a	2 find / f	3 tag [t	4 permi ssion @ p	5 mana ge > m	6 verify ! v	7 quit . q

표 1: 올바른 모든 명령어들 (명령어군별)

해석: 명령어는 이미 문법 형식에 구분되어 쓰여있으므로 해석 규칙이 필요 없습니다. <단어열[1]>은 (만일 있다면) 그 속의 단어들 각각을 인자들로 간주합니다.

의미 규칙: <명령어>자리에 어떤 명령어군에 속한 명령어가 들어오는지에 따라서, 위의 <단어열[1]>이 허용되는지 여부와 (허용된다면) 그 속에 어떤 인자 몇 개가 쓰일 수 있는지도 달라집니다 표 2는 각 명령어군 별 인자 규칙과 설명입니다.

명령어군				올바른 인자들	설명
1	add	a	+	한 개 이상	추가 메뉴를 출력합니다.
2	find	f	/	없거나 여러 개	최근 내역을 출력하거나 인자의 날짜(혹은 구간), 태그로 내역을 검색합니다.
3	tag	t	[	없음	태그 메뉴를 출력합니다.
4	permission	p	@	없음	권한 메뉴를 출력합니다.

5	manage	m	>	없음	전체 계좌 목록과 계좌 메뉴를 출력합니다.
6	verify	v	!	없음	데이터 무결성 확인 및 처리를 진행합니다.
7	quit	q	.	없음	프로그램을 종료합니다.

표 2: 명령어군별 대표 명령어들과 인자 규칙 및 설명

문법 오류 결과: 만일 주 프롬프트에서 그냥 곧바로 Enter키만 누르거나 (즉, 빈 문자열 입력), 공백(들)만 입력하거나, 입력 중 첫번째 단어가 명령어가 아닐 경우, 틀린 입력으로 간주하고 ("잘못된 입력입니다." 같은 진부한 안내 없이, 조용히) 표 1에 준하는 표준 명령어 및 인자 안내 화면을 출력하고 주 프롬프트로 되돌아갑니다. 프로그램의 사용 흐름 상, 이 출력은 통상적인 '메뉴 번호 선택' 방식으로 진행되는 프로그램의 '메인 메뉴'에 해당합니다.

```
AccountNumber > 뭘 입력해야 하지?↵
=============================================================
     명령어군              올바른인자들              설명
-------------------------------------------------------------------------------------
1  add        a    +          한 개 이상              추가 메뉴를 출력~
-------------------------------------------------------------------------------------
2  find       f    /          없거나 여러 개          최근 내역을 출력~
-------------------------------------------------------------------------------------
3  tag        t    [            없음                  태그 메뉴를 출력~
-------------------------------------------------------------------------------------
4  permisson  p    @            없음                  권한 메뉴를 출력~
-------------------------------------------------------------------------------------
5  manage     m    >            없음                  전체 계좌 목록을~
-------------------------------------------------------------------------------------
6  verify     f    !            없음                   데이터 무결성~
-------------------------------------------------------------------------------------
7  quit       q    .            없음                  프로그램을 종료~
-------------------------------------------------------------------------------------
AccountNumber >
```

위의 창에서 설명란은 표의 공간이 부족하여 생략하였습니다. 내용은 표 2와 같습니다.

의미 오류 혹은 정상 결과 : 만일 입력 중 첫번째 단어가 명령어가 맞으면, 표 의 의미 규칙을 기준으로 의미 오류 혹은 정상으로 판정합니다. 두 경우 모두 프로그램의 구체적인 동작 방식은 이후의 각 명령어군별 소절에서 명시하겠습니다.

7.1 종료 명령어군

프로그램을 종료합니다.

문법 규칙 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

의미 규칙: 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다.

비정상 결과 : 만일 문법 형식에 위배되면 (인자가 존재하면) 문법에 맞지 않는다는 오류 메세지만 출력하고 주 프롬프트를 다시 출력합니다.

```
AccountNumber > quit now ↲
. ! ! 오류: 인자가 없어야 합니다.

AccountNumber >
```

정상 결과: 명령이 올바르게 입력되면 즉시 프로그램을 종료합니다.

```
AccountNumber > quit ↲

PS D: ₩>
```

7.2 무결성 확인/처리 명령어군

데이터 파일의 무결성을 확인하고 처리합니다.

문법 규칙 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

의미 규칙 : 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다.

비정상 결과 : 만일 문법 형식에 위배되면(인자가 존재하면) 문법에 맞지 않다는 오류 메세지만 출력하고 주 프롬프트를 다시 출력합니다.

정상 결과: 문법에 맞게 명령이 입력되면 (6.2절의 규칙에 따라)의 주 프롬프트로 다시 돌아갈 수도 있고 프로그램이 종료될 수도 있습니다.

7.3 추가 명령어군

수입, 지출 내역을 추가합니다. 이 때 태그는 추가 명령어에 인자들로 바로 전달하며, 이는 원하는 태그가 없어 추가를 원할 경우 바로 주프롬프트로 돌아가 태그 명령어군을 사용할 수 있도록 하기 위함입니다.

문법 규칙 : 이 명령은 태그를 입력받기 위해 인자가 1개 이상이어야 하며, 하나의 태그가 공백을 허용하기 때문에, 즉 하나의 인자로만 구성되지 않기 때문에 추가 명령어군에 대해서만 7절의 문법 형식에서의 인자로의 해석 전의 <단어열[1]>에 대해 문법 형식을 정합니다. 5.3절의 태그 문법 형식에 맞는 문자열을 <태그>라고 할 때, 문법적으로 올바른 <단어열[1]>은 다음 두 형식 중 하나입니다:

- **<비개행공백열^0><태그><비개행공백열^0>**
- **<비개행공백열^0><숫자>.<숫자><비개행공백열^0>**

의미 규칙 : 입력된 인자가 1개 이상이고, <태그>가 입력된 경우 <태그>는 5.3절의 태그 의미 규칙에 따라야 하며, 이에 따라 처리 및 수정된 내용은 기존 하위 태그들 중 하나와 동치되어야 합니다. 입력된 인자가 1개 이상이고, <숫자>.<숫자>가 입력된 경우 태그 목록에 존재하는 하위 태그 위치 중 하나와 동치되어야 합니다.

비정상 결과 : 문법 형식 또는 의미 규칙에 어긋나는 경우

1. 문법 형식에 위배될 경우 중 인자가 없는 경우 태그 목록과 오류 메세지를 출력하고, 주 프롬프트로 돌아갑니다.

```
AccountNumber > add↵
.!! 오류: 추가 명령어 뒤에 하나의 [태그]나 태그 위치를 입력해야 합니다.

=======================태그 목록 출력=======================
1[음식]
 ㄴ1.1[카페]
 ㄴ1.2[식사]
 ㄴ1.3[술]
2[선물]
 ㄴ2.1[애인]
 ㄴ2.2[가족]

AccountNumber >
```

2. 문법 형식에 위배된 경우 중 1이상, '태그 목록에 존재하는 상위 태그의 개수' 이하의 정수인 <숫자>가 입력된 경우, 또는 의미 규칙에 위배된 경우 중 기존 상위 태그들 중 입력된 <태그>와 동치되는 태그가 존재할 경우, 해당 <숫자>와 동치되는 태그 위치의 상위 태그 또는 해당 <태그>와 동치되는 상위 태그의 하위 태그 목록과 오류 메시지를 출력하고, 주 프롬프트로 돌아갑니다.

```
AccountNumber > add [음식]↵
또는
AccountNumber > add 1↵
..! 상위 태그입니다. 하위 태그를 입력해 주세요.

====================[음식] 하위 태그 목록 출력====================
1[음식]
 ㄴ1.1[카페]
 ㄴ1.2[식사]
 ㄴ1.3[술]

AccountNumber >
```

3. 위의 모든 경우를 제외하고, 의미 규칙에 위배된 경우 의미 규칙에 맞지 않는다는 오류 메세지와 태그 목록, 메인 프롬프트의 태그 명령어를 출력하고 주 프롬프트로 돌아갑니다.

```
AccountNumber > add [동아리]↵
..! 존재하지 않는 태그입니다. 태그 추가 및 관리는 메인 메뉴에서 tag, t, [ 로 열 수 있습니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
AccountNumber > add 4.1↵
..! 존재하지 않는 태그 위치입니다. 태그 추가 및 관리는 메인 메뉴에서 tag, t, [ 로 열 수 있습니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
AccountNumber >
```

4. 위의 모든 경우를 제외하고, 문법 형식에 위배된 경우 문법 형식에 맞지 않는다는 오류 메세지만 출력하고 주 프롬프트로 돌아갑니다. 이 때, <숫자>로 시작되는 경우, 무조건 태그 위치를 입력하려 했던 것으로 판단하고, 이외의 경우는 <태그>를 입력하려 했던 것으로 판단합니다. (아래 예시의 '5'를 입력한 경우 참고)

```
AccountNumber > add [식사⇥친구]↵
.!! 오류: 태그는 탭과 개행 문자의 포함을 허용하지 않습니다.

AccountNumber > add [식사][친구]↵
.!! 오류: 추가 명령어 뒤에 하나의 [태그]를 입력해야 합니다.

AccountNumber > add 친구↵
.!! 오류: 태그를 '[', ']'로 감싸야 합니다.

AccountNumber > add 1..1↵
또는
AccountNumber > add 5↵
.!! 오류: 태그 위치는 <숫자>.<숫자>로만 입력 가능합니다.

AccountNumber >
```

정상 결과 : 하위 태그와 동치되는 <태그>나 하위 태그 위치와 동치되는 태그 위치가 입력되면 다음 단계인 내역 입력 단계로 넘어갑니다.

7.3.1 부 프롬프트 1 : 내역 입력

태그를 올바르게 입력하고 나면, 내역을 입력하는 부 프롬프트가 출력됩니다. 이 때 선택된 하위 태그를 상위 태그와 함께 출력합니다.

```
AccountNumber > add [카페]↵
또는
AccountNumber > add 1.1↵
```

```
AccountNumber > [음식][카페] 내역>
```

문법 규칙 : 이 명령은 인자가 1개 혹은 2개여야 하고, 1개일 경우 금액의 문법 형식에, 2개일 경우 첫번째 인자는 5.2절의 금액의 문법 형식에, 두번째 인자는 5.1절의 날짜의 문법 형식에 부합해야 합니다.

해석 방법 : 첫번째 인자는 금액으로, 두번째 인자는 날짜로 해석합니다. 두번째 인자가 없을 경우 날짜는 입력 시점으로 해석합니다.

의미 규칙 : 날짜는 입력 시점보다 미래 시점일 수 없습니다.

비정상 결과 : 이름내역이 문법 형식이나 의미 규칙에 위배되면 그에 상응하는 오류 메세지를 출력하고 내역 입력 부 프롬프트를 다시 출력합니다. “금액, 날짜 순서로 입력해 주세요. 날짜만 생략할 수 있습니다.”는 인자가 2개 이하인 경우 모든 오류에 대해 함께 출력합니다.

```
AccountNumber > [음식][카페] 내역> 2021.01.02 2000원↵
.!! 오류: 금액, 날짜 순서로 입력해 주세요. 날짜만 생략할 수 있습니다.
금액은 ',', '원', 숫자로만 써주세요.
날짜는 '-', '/', '.', 숫자로만 써주세요.

AccountNumber > [음식][카페] 내역> 2021. 01. 02 -2000원↵
.!! 오류: 금액, 날짜 순서로 입력해 주세요. 날짜만 생략할 수 있습니다.
.!! 오류: 인자가 너무 많습니다. 날짜와 금액은 공백을 허용하지 않습니다.

AccountNumber > [음식][카페] 내역> +2000원 2021.01.03↵
.!! 오류: 금액, 날짜 순서로 입력해 주세요. 날짜만 생략할 수 있습니다.
.!! 오류: 미래의 내역은 작성할 수 없습니다.

AccountNumber > [음식][카페] 내역 >
```

정상 결과: 만일 입력 내용이 문법 형식과 의미 규칙에 맞다면, 선택한 태그, 입력받은 금액과 날짜를 다시 출력하고 다음 단계인 저장 확인 단계로 넘어갑니다. 이 때, 날짜를 생략한 경우 날짜 뒤에 "(오늘)"이라고 함께 출력합니다. "(오늘)"은 입출금 내역을 저장할 때는 당연히 저장되지 않습니다.

7.3.2 부 프롬프트 2 : 저장 확인

선택한 태그, 입력받은 금액과 날짜를 다시 출력하고 저장할지 여부를 묻습니다. 이 때, 날짜를 생략한 경우 날짜 뒤에 "(오늘)"이라고 함께 출력합니다. "(오늘)"은 입출금 내역을 저장할 때는 당연히 저장되지 않습니다.

```
AccountNumber > [음식][카페] 내역> 2000↵
입력 내용: [음식][카페] +2000원 2021.01.02 (오늘)
AccountNumber> 정말 저장하시겠습니까? (.../No) >
```

문법 규칙 : 어떤 키 입력 문자열도 모두 문법적으로 올바릅니다.

해석 : 키 입력 문자열이 “No”라는 글자에 정확히 (앞 · 뒤 공백도 없이, 대 · 소문자도 똑같이) 일치해야 부정으로 해석하며, 그 외의 모든 입력은 (그냥 ↵ 만 눌러 입력한 빈 문자열도 포함해서) 전부 긍정으로 해석합니다.

의미 규칙 : 별도의 의미 규칙은 없습니다.

비정상 결과 : 이 입력의 결과로는 어떤 경우에도 비정상 결과가 나오지 않습니다.

정상 결과 : 입력한 답을 해석한 결과가 긍정이면, 지금까지 입력했던 입출금 내역을 레코드 형식의 문자열로 만들어서 파일 맨 끝에 새 줄로 추가하여 저장합니다. 부정이면, 지금까지 입력했던 입출금 내역을 그냥 폐기하고 주 프롬프트로 돌아갑니다.

7.4 검색 및 수정 명령어군

문법 형식 : 이 명령은 인자가 없거나 있을 수 있고, (만일 있다면) 각 인자 또는 인자들(태그는 여러개의 인자로 구성되므로)은 날짜와 태그 둘 중 하나의 문법형식을 따라야 합니다. 단, 날짜의 경우 4자리 또는 6자리로 작성하는 것이 허용됩니다. (5.1절 참고) 날짜의 문법형식을 따르는 인자는 2개를 초과할 수 없고, 태그의 문법형식을 따르는 인자는 개수에 제한이 없습니다. 이는 태그 하나에 여러개의 인자를 포함하는 것뿐만 아니라, 여러개의 태그를 작성하는 것도 포함해 허용하는 것입니다.

해석 : 날짜의 문법형식을 따르는 인자는 날짜로, 태그의 문법형식을 따르는 인자들은 태그로 해석합니다. 인자가 없는 경우에도 날짜에 대한 검색으로 해석합니다.

의미 규칙 : 인자가 있는 경우, 날짜는 날짜의 의미 규칙을, 태그는 태그의 의미 규칙을 따라야 합니다.

비정상 결과 : 입력된 내용이 문법 형식이나 의미 규칙을 위배하는 경우, 오류 메세지를 출력하고 주 프롬프트로 돌아갑니다.

```
AccountNumber > find 2019 2020 2021↵
..! 날짜는 2개까지만 입력할 수 있습니다.

AccountNumber > find 음식
.!! 오류: 태그를 '[', ']'로 감싸야 합니다.

AccountNumber >
```

정상 결과 : 문법 형식과 의미 규칙을 위배하지 않을 경우, 정상결과는 다음과 같습니다.

1. 날짜만 입력되었거나 태그만 입력된 경우, 입력 내용을 '검색' 했을 때 합치되는 데이터 요소가 들어있는 모든 레코드들을 찾아서 날짜 순, 태그 위치 순, 레코드 위치 (행) 순으로 정렬하여 출력합니다. 이때 사용자가 읽기 권한만 가지고 있을 경우는 주 프롬프트로 돌아가고, 그렇지 않은 경우(즉, 항목 수정 권한이 있음)에는 내역 선택 단계로 들어갑니다. 만약 합치되는 내용, 즉 검색 결과가 없으면 사용자의 권한과 무관하게 주 프롬프트로 돌아갑니다. 이 때, 각 데이터 요소별로 ‘검색

시 무엇을 합치된 것으로 간주할 것인지'가 다른데, 이는 5.1절의 "검색", 5.3절의 "동치 비교" 항목에 각각 명시되어 있습니다.

```
AccountNumber > find [카페]↵
1 [음식][카페] -2000원 2021.01.02
```

```
AccountNumber > find 202101↵
1 [음식][카페] -2000원 2021.01.02
2 [선물][가족] -10000원 2021.01.02
```

```
AccountNumber > find [음식]↵
1 [음식][카페] -2000원 2021.01.02
2 [음식][식사] -6000원 2021.03.31
```

```
AccountNumber > find [음식] [선물]↵
1 [음식][카페] -2000원 2021.01.02
2 [선물][가족] -10000원 2021.01.02
5 [음식][식사] -6000원 2021.03.31
```

```
AccountNumber > find↵ (총 내역이 3개인 상황)
1 [음식][카페] -2000원 2021.01.02
2 [선물][가족] -10000원 2021.01.02
5 [음식][식사] -6000원 2021.03.31
```

```
AccountNumber > find [술]↵
.!! 검색 결과가 없습니다.

AccountNumber >
```

2. 날짜와 태그가 함께 입력된 경우, 1과 같이 검색한 결과들의 교집합만을 1과 동일한 순서로 정렬하여 출력하고, 사용자의 권한에 따라 1과 동일하게 동작합니다.

```
AccountNumber > find 2021/01 [카페]↵
1 [음식][카페] -2000원 2021.01.02
```

7.4.1 부 프롬프트 1: 내역 선택

문법 규칙 : 하나의 인자만을 허용하며, 검색된 내역들의 개수 이하의 음이 아닌 정수여야 합니다.

의미 규칙 : 이 명령어는 별도의 의미 규칙이 없습니다.

비정상 결과 : 입력된 값이 문법 형식에 위배될 경우, 오류 메세지를 출력하고 내역 선택으로 돌아갑니다.

```
AccountNumber > 내역 선택(취소는 0)> 3↵
..! 존재하지 않는 번호입니다. 범위 내의 정수만 입력할 수 있습니다.

AccountNumber> 내역 선택 (취소는 0)>
```

정상 결과 : 0이 입력된 경우, 주프롬프트로 돌아갑니다. 양의 정수가 입력된 경우, 입력된 번호에 해당하는 내역을 선택하고, 선택된 내역을 출력한 뒤, 검색 후 작업 선택 단계로 넘어갑니다.

7.4.2 부 프롬프트 2 : 검색 후 작업 선택

문법 형식 : 하나의 인자만을 허용하며, 1 이상 3 이하의 정수여야 합니다.

의미 규칙 : 이 명령어는 별도의 의미 규칙이 없습니다.

비정상 결과 : 입력된 값이 문법 형식에 위배될 경우, 오류 메세지와 선택된 내역을 (재)출력하고 검색 후 작업 선택으로 돌아갑니다.

```
AccountNumber > 내역 선택(취소는 0)> 1↵
선택된 내역: [음식][카페] -2000원 2021.01.02
..! 원하시는 기능을 입력하세요(숫자 하나)
    1. 내역 수정          2. 내역 삭제          3. 취소

AccountNumber > 검색 후 작업 > 4↵
..! 1~3 중 원하는 기능의 메뉴 번호를 입력하세요.
    1. 내역 수정          2. 내역 삭제          3. 취소
선택된 내역: [음식][카페] -2000원 2021.01.02

AccountNumber > 검색 후 작업 >
```

정상 결과 : 1을 입력한 경우 내역 수정 부 프롬프트3으로, 2를 입력한 경우 내역 삭제 부 프롬프트 5로 이동합니다. 3을 입력한 경우 내역 선택으로 돌아갑니다.

7.4.3 부 프롬프트 3 : 내역 수정

문법 규칙 : 내역 수정에서 올바른 키 입력은 다음과 같습니다.

- <단어> <비개행공백열[1]> <단어열[1]>

단, 위의 <단어> 자리에는 반드시 tag, date, money 중 하나만을 써야 하며, 이는 7절 주 프롬프트의 '명령어'들처럼 이들만이 문법적으로 올바릅니다.

의미 규칙 : <단어열[1]>은 <단어>에 따라 문법 형식과 의미 규칙이 결정됩니다.

문법 오류 결과 : 만일 곧바로 ↵ 만 누르거나 (즉, 빈 문자열 입력), 공백(들)만 입력하거나, 입력 중 첫번째 단어가 tag, date, money 중 하나가 아닐 경우, 틀린 입력으로 간주하고 입력 예시 안내 화면을 출력한 뒤 내역 수정 부 프롬프트로 돌아갑니다.

```
AccountNumber > 검색 후 작업> 1↵
AccountNumber > 내역 수정> ↵
```

..! 수정하고 싶은 항목에 따라 tag, date, money와 수정할 내용을 입력하세요.
입력 예시) tag [카페]
date 2021년 01월 02일
money -1000원

AccountNumber > 내역 수정>

의미 오류 혹은 정상 결과 : 구체적인 동작 방식은 다음과 같습니다.

1. <단어>가 tag일 때 : <단어열[1]>은 7.3절의 추가 명령어군에서의 <단어열[1]>과 동일한 문법 형식과 의미 규칙을 따릅니다.

비정상 결과 : 주 프롬프트가 아닌 내역 수정 부 프롬프트 3으로 돌아가는 것을 제외하면 오류 메세지를 포함하여, 7.3절에서와 동일하게 작동합니다.

정상 결과 : 선택한(수정할) 내역의 태그를 입력된 하위태그로 수정했을 때의 내역을 출력한 뒤, 다음 단계인 수정 확인으로 넘어갑니다.

AccountNumber > 내역 수정> tag [동아리]↵
..! 존재하지 않는 태그입니다. 태그 추가 및 관리는 메인 메뉴에서 tag, t, [로 열 수 있습니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)

AccountNumber > 내역 수정>

2. <단어>가 date일 때: <단어열[1]>은 7.3.1절의 추가 명령어군의 내역입력 부 프롬프트 1에서의 두번째 인자와 동일한 문법 형식과 의미 규칙을 따릅니다.

비정상 결과 : 입력이 문법 형식이나 의미 규칙에 위배되면 그에 상응하는 오류 메세지를 출력하고 내역 수정 부 프롬프트 3으로 돌아갑니다.

정상 결과 : 선택한(수정할) 내역의 날짜를 입력된 날짜로 수정했을 때의 내역을 출력한 뒤, 다음 단계인 수정 확인으로 넘어갑니다. 이 때, 날짜를 생략한 경우 날짜 뒤에 "(오늘)"이라고 함께 출력합니다. "(오늘)"은 입출금 내역을 저장할 때는 당연히 저장되지 않습니다.

AccountNumber > 내역 수정> date 2021.1.2↵
..! 날짜는 연도 4자리, 월 2자리, 일 2자리로 입력해야 합니다.

AccountNumber > 내역 수정>

3. <단어>가 money일 때 : <단어열[1]>은 7.3.1절의 추가 명령어군의 내역입력 부 프롬프트 1에서의 첫번째 인자와 동일한 문법 형식과 의미 규칙을 따릅니다.

비정상 결과 : 입력이 문법 형식이나 의미 규칙에 위배되면 그에 상응하는 오류 메세지를 출력하고 내역 수정 부 프롬프트 3으로 돌아갑니다.

정상 결과 : 선택한(수정할) 내역의 금액을 입력된 금액으로 수정했을 때의 내역을 출력한 뒤, 다음 단계인 수정 확인으로 넘어갑니다.

7.4.4 부 프롬프트 4 : 수정 확인

7.3.2절의 저장 확인과 동일한 문법 형식, 의미 규칙, 해석을 따르고, 비정상 결과도 동일합니다.

정상 결과 : 입력한 답을 해석한 결과가 긍정이면, 입력된 수정값을 파일의 기존 입출금내역 레코드에 반영합니다. 부정이면, 지금까지 입력했던 내역을 그냥 폐기하고 주 프롬프트로 돌아갑니다.

```
AccountNumber > 내역 수정> tag [식사]↵
수정된 내역: [음식][식사] -2000원 2021.01.02

AccountNumber> 정말 수정하시겠습니까? (.../No) >
```

7.4.5 부 프롬프트 5: 내역 삭제

7.3.2절의 저장 확인과 동일한 문법 형식, 의미 규칙을 따르고, 비정상 결과도 동일합니다.

해석 : 키 입력 문자열이 "Yes" 라는 3 글자에 정확히(앞뒤 공백 없이, 대소문자도 똑같이) 일치해야 긍정으로 해석하며, 그 외의 모든 입력은(↵만 눌러 입력한 빈 문자열도 포함해서) 전부 부정으로 해석합니다.

정상 결과 : 입력한 답을 해석한 결과가 긍정이면, 특정한 레코드를 데이터 파일에서 삭제한 후 6.2절의 파일 무결성 확인/처리를 수행하고, 그 결과에 따라 이후 단계를 진행합니다. 부정이면, 그냥 주 프롬프트로 돌아갑니다.

```
AccountNumber > 검색 후 작업 > 2↵

AccountNumber> 정말 삭제하시겠습니까? (Yes/...) >
```

7.5 태그 편집 명령어군

태그 종류를 수정하거나 삭제합니다.

문법 형식 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

의미 규칙 : 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다

비정상 결과 : 만일 문법 형식에 위배되면(인자가 존재하면) 문법에 맞지 않다는 오류 메세지만 출력하고 주 프롬프트를 다시 출력합니다.

```
AccountNumber > tag eat↵
.! ! 오류: 인자가 없어야 합니다.

AccountNumber >
```

정상 결과 : 문법에 맞게 명령이 입력되면 태그 메뉴가 출력 됩니다. 이에 관해서는 5.3.1 소절에서 부연합니다. 태그 이름 입력 및 추가 시 규칙은 5.3절에 명시되어 있습니다.

```
AccountNumber > tag↵

(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
..! 원하시는 기능을 입력하세요(숫자 하나)
        1. 태그 추가          2. 태그 수정          3. 태그 삭제

AccountNumber >
```

바로 위의 프롬프트에서의 문법 규칙과 의미 규칙은 7.4.2 부 프롬프트 2: 검색 후 작업 선택 프롬프트의 문법 규칙과 의미 규칙과 동일합니다.

비정상 결과 : 입력된 값이 문법 형식에 위배될 경우, 오류 메시지를 출력하고 해당 프롬프트를 다시 띄워서 재입력을 받습니다.

정상 결과 : 1을 입력한 경우 7.5.1 부 프롬프트1으로, 2를 입력한 경우 7.5.2 부 프롬프트2로 이동합니다. 3을 입력한 경우 7.5.3 부 프롬프트3으로 이동합니다.

7.5.1 부 프롬프트 1 : 태그 추가

태그를 태그 목록에 추가합니다.

문법 형식 : 5.3 절의 태그 문법 규칙에 맞는 문자열을 <태그>이라고 하고 태그 목록의 위치를 <위치>이라고 할 때, 문법적으로 올바른 키 입력 문자열은 다음 형식에 부합하는 문자열입니다

- <비개행공백열0><위치><비개행공백열^0><태그><비개행공백열^0>

의미 규칙 : 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다.

검사 항목1 : 해당 위치에 이미 태그가 존재하는지 확인합니다.

검사 항목2 : 태그 목록 중에 입력한 태그 이름과 '동치'인 태그 이름이 있는지 확인합니다.

비정상 결과 : 만일 문법 형식에 위배되면 (인자가 존재하면) 문법에 맞지 않는다는 오류 메세지만 출력하고 태그 추가를 다시 실행합니다. 만일 해당 태그 위치를 사용할 수 없으면 태그 위치를 사용할 수 없다는 메세지만 출력하고 태그 추가를 다시 실행합니다. 해당 태그 이름이 이미 존재하면 이미 존재한다는 메세지만 출력하고 태그 추가를 다시 실행합니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 4 [연⇥구⇥비] ↲
. ! ! 오류 : 태그 이름에는 탭이나 개행 문자가 포함될 수 없습니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 4 [선물] ↲
. ! ! 오류 : 해당 이름의 태그가 이미 존재 합니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 2 [선물] ↲
. ! ! 오류 : 해당 위치에 태그가 이미 존재 합니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 4 [*******5224*******] ↲
.!! 오류 : 태그 이름으로 사용할 수 없습니다. 아래 사항을 참고해주세요.
1. 공백 포함 1520글자의 문자열을 입력해주세요.
2. 숫자와 특수문자들로만 이루어진 문자열은 불허입니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 4 [500] ↲
.!! 오류 : 숫자와 특수문자들로만 이루어진 문자열은 불허입니다.

... 추가할 위치와 태그 이름을 입력하세요
ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber >

정상 결과 : 태그 추가가 완료되었음을 알리고 태그를 태그 목록에 추가합니다.

... 추가할 위치와 태그 이름을 입력하세요

ex) 주태그일시 >4 [운동]　　부태그일시 > 3.4 [방탈출]

AccountNumber > 4 [운동]↲
... 태그 추가가 완료되었습니다.

AccountNumber >

7.5.2 부 프롬프트 2 : 태그 수정

문법 형식 : 5.3 절의 태그 문법 규칙에 맞는 문자열을(태그)이라고 하고 태그 목록의 위치를<위치>이라고 할 때, 문법적으로 올바른 키 입력 문자열은 다음 형식에 부합하는 문자열입니다.

- **<비개행공백열0><위치><비개행공백열[1]><태그>**

의미 규칙: 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다

검사 항목1 : 해당 위치에 태그가 존재하는지 확인합니다.

검사 항목2 : 태그 목록 중에 입력한 태그 이름과 '동치'인 태그 이름이 있는지 확인합니다.

비정상 결과 : 만일 문법 형식에 위배되면 (인자가 존재하면) 문법에 맞지 않는다는 오류 메세지만 출력하고 태그 수정을 다시 실행합니다. 만일 해당 태그 위치가 없으면 태그 위치가 존재하지 않는다는 메세지만 출력하고 태그 추가를 다시 실행합니다. 해당 태그 이름이 이미 존재하면 이미 존재한다는 메세지만 출력하고 태그 추가를 다시 실행합니다.

... 수정할 위치와 태그 이름을 입력하세요

AccountNumber > 2 [연⇥구⇥비] ↲
. ! ! 오류 : 태그 이름에는 탭이나 개행 문자가 포함될 수 없습니다.

... 수정할 위치와 태그 이름을 입력하세요

AccountNumber > 3.1 [오락] ↲
. ! ! 오류 : 해당 이름의 태그가 이미 존재 합니다.

... 추가할 위치와 태그 이름을 입력하세요

AccountNumber > 99 [선물] ↲
. ! ! 오류 : 해당 위치에 태그가 존재하지 않습니다.

... 추가할 위치와 태그 이름을 입력하세요

AccountNumber >

정상 결과 : 태그 추가가 완료되었음을 알리고 태그를 태그 목록에 추가합니다.

AccountNumber > 2↵
... 수정할 위치와 태그 이름을 입력하세요
ex) > 1 [식사]

AccountNumber > 1.2 [디저트]↵
... 태그 수정이 완료되었습니다.

AccountNumber >

7.5.3 부 프롬프트 3: 태그 삭제

문법 형식 : 주 태그일 경우 해당 위치의 숫자 하나만 적고 부 태그일 경우 <숫자><.><숫자>의 형식으로 입력합니다.

의미 규칙 : 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다

검사 항목1 : 해당 위치에 태그가 존재하는지 확인합니다.

검사 항목2 : 주 태그일 경우 하위 태그가 존재하는지 확인합니다.

비정상 결과 : 만일 문법 형식에 위배되면 (인자가 존재하면) 문법에 맞지 않는다는 오류 메세지만 출력하고 태그 삭제을 다시 실행합니다. 만일 해당 태그 위치가 없으면 태그 위치가 존재하지 않는다는 메세지만 출력하고 태그 삭제를 다시 실행합니다. 해당 태그가 주 태그이고 하위 태그가 존재한다면 삭제 할 수 없다는 메세지만 출력하고 태그 삭제를 다시 실행합니다.

(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
... 삭제할 태그 위치를 입력하세요.

AccountNumber > 2-3 ↵
. ! ! 오류 : 태그 위치에는 숫자와 . 만 입력 가능합니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
... 삭제할 태그 위치를 입력하세요.

AccountNumber > 3.11 ↵
. ! ! 오류 : 해당 위치에 태그가 존재하지 않습니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
... 삭제할 태그 위치를 입력하세요.

AccountNumber > 3 ↵
. ! ! 오류 : 해당 위치에 태그에 하위 태그들이 존재하면 삭제 할 수 없습니다.
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
... 삭제할 태그 위치를 입력하세요.
AccountNumber >

정상 결과 : 태그 추가가 삭제되었음을 알리고 태그 목록에서 해당 위치의 태그를 삭제합니다.

```
AccountNumber > 3↵
(태그 목록이 출력되는 부분은 7.3절의 예시와 동일하여 생략합니다.)
... 삭제할 태그 위치를 입력하세요.
... 주태그일 경우 하위 태그가 존재할 시 삭제 불가합니다.
ex) 3.1
AccountNumber > 3.1↵
... 태그 삭제가 완료되었습니다.

AccountNumber >
```

7.6 권한 변경 명령어군

관리자가 공용 계좌 사용자들 혹은 공용 계좌 접근 권한을 요청한 사용자들의 권한을 변경합니다.

문법 규칙 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

의미 규칙: 이 명령은 문법 형식 외에 더 준수해야 할 의미 규칙이 없습니다.

비정상 결과: 만일 문법 형식에 위배되면(인자가 존재하면) 문법에 맞지 않는다는 오류 메세지와 대표 명령어를 출력한 뒤, 주 프롬프트를 다시 출력합니다.

```
AccountNumber > 4 @ ↵
. ! ! 오류: 인자가 없어야 합니다.

AccountNumber >
```

정상 결과: 문법에 맞게 명령이 입력되면:

1. 공용 계좌 사용자멤버 목록을 출력합니다. 공용 계좌 사용자 권한 요청을 보냈으나 관리자가 권한을 부여하지 않은 경우는 권한 자리가 '공백'으로 출력됩니다.
2. 첫번째 부 프롬프트를 출력하고 사용자 이름을 입력받습니다. 이에 관해서는 7.6.1 소절에서 부연합니다.
3. 입력받은 이름과 '동치'인 레코드가 없는지 확인해서, 만일 없으면 "사용자 이름이 공용 계좌 목록에 존재하지 않습니다."라는 오류 메세지를 출력하고 다시 이름 입력을 받을 수 있도록 출력합니다.
4. 두 번째 부 프롬프트를 출력하고 권한 메뉴를 입력받습니다. 이에 관해서는 7.6.2 소절에서 부연합니다.
5. 사용자의 권한과 선택한 권한이 서로 다른지 확인하여 만일 같으면 "이미 가지고 있는 권한입니다."라는 오류 메세지를 출력하고 다시 권한 메뉴를 입력 받습니다.
6. 세번째 부 프롬프트를 출력하고 저장 여부를 최종 확인합니다. 이에 관해서는 7.6.3 소절에서 부연합니다.

7.6.1 부 프롬프트 1: 사용자 이름 입력

공용 계좌 사용자 목록을 출력한 뒤 사용자 이름을 키 입력하는 부 프롬프트가 출력됩니다:

```
AccountNumber > 4 ↲
공용 계좌 사용자 목록
유병헌 관리자 권한
구선민 수정 및 삭제 권한
정세훈 읽기만 가능
김정우
이윤정 읽기만 가능

AccountNumber: 사용자 이름 >
```

문법 형식: 문법적으로 올바른 키 입력 문자열은 공백과 개행이 없는 순수 문자열입니다.

의미 규칙: 이름은 공용 계좌 목록 파일에 존재해야합니다.

검사 항목: 데이터 파일에 사용자가 포함되어 있는지 확인합니다.

비정상 결과 : 이름이 문법 형식에 위배거나 파일에 사용자가 존재하지 않으면 그에 상응하는 오류 메세지를 출력하고 사용자 이름 입력 부 프롬프트를 다시 출력합니다.

```
AccountNumber: 사용자 이름 > ↲
. ! ! 오류: 사용자 이름은 빈 문자열일 수 없습니다.

AccountNumber: 사용자 이름 >  ␣⇥␣↲
. ! ! 오류: 사용자 이름은 (각종) 공백들로만 구성될 수 없습니다.

AccountNumber: 사용자 이름 > 유⇥병헌↲
. ! ! 오류: 사용자 이름의 유효한 첫 글자와 유효한 마지막 글자 사이에는 탭이 들어갈 수 없습니다.

AccountNumber: 사용자 이름 > 솜사탕↲
. ! ! 오류: 사용자 이름이 공용 계좌 목록에 존재하지 않습니다.

 AccountNumber: 사용자 이름 >
```

정상 결과 : 인자가 문법 형식 및 의미 규칙에 부합하면 권한 메뉴를 출력하고 다음 단계인 권한 메뉴 입력 단계로 넘어갑니다.

7.6.2 부 프롬프트 2: 권한 입력

사용자 이름을 올바르게 입력하고 검사 항목을 확인/처리하고 나면, 권한 메뉴가 출력되고 권한 입력을 키 입력하는 부 프롬프트가 출력됩니다. 이때 문법 규칙과 의미 규칙은 7.4.2절의 검색 후 작업 선택 부프롬프트에서와 동일합니다.

```
권한 메뉴
1. 관리자 권한
2. 수정 및 삭제 권한
```

```
3. 읽기만 가능

AccountNumber : 권한 입력 >
```

검사항목1 : 공용 계좌 사용자 목록에서 사용자 권한이 입력받은 사용자와 동일하지 않은지 확인합니다.

비정상 결과 : 문법 형식에 부합하지 않거나 기존 권한과 입력받은 권한이 같으면 그에 상응하는 오류 메세지를 출력하고 권한 입력 부 프롬프트를 다시 출력합니다.

```
AccountNumber : 권한 입력 > 1.0↵
. ! ! 오류: 소수점이 존재하지 않는 양의 정수여야 합니다.

AccountNumber : 권한 입력 > 8↵
. ! ! 오류: 권한 메뉴 번호가 아닙니다.

AccountNumber : 권한 입력 > 2↵
. ! ! 오류: 기존 권한과 동일합니다.

AccountNumber : 권한 입력 >
```

정상 결과 : 입력받은 권한이 기존 권한과 다르면 다음 단계인 저장 확인 단계로 넘어갑니다.

7.6.3 부 프롬프트 3: 저장 확인

권한까지 입력하고 나면, 지금까지 입력한 사용자 이름, 변경할 권한을 다시 출력하고 저장할지 여부를 묻습니다.

7.3.2절의 저장 확인과 동일한 문법 형식, 의미 규칙, 해석을 따르고, 비정상 결과도 동일합니다.

```
AccountNumber : 권한 입력 > 읽기만 가능↵
. . . 현재까지 입력한 내역입니다:
=> 사용자 이름: 구선민
=> 변경 권한: 읽기만 가능

AccountNumber : 정말 저장하시겠습니까? (.../No) >
```

정상 결과: 입력한 답을 해석한 결과가 :

- ■ 긍정이면, 파일에 존재하는 사용자의 권한을 입력받은 권한으로 변경한 후 저장합니다.
- ■ 긍정이고, 관리자 권한으로 변경하면 기존 관리자 권한을 가진 사용자의 권한은 수정 및 삭제 권한으로 변경됩니다.
- ■ 부정이면, 지금까지 입력했던 내역을 그냥 폐기하고 주 프롬프트로 돌아갑니다.

7.7 계좌 선택 명령어군

인자에 따라 계좌 목록을 출력하거나 해당 계좌를 선택합니다.

인자 문법 형식 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

인자 의미 규칙 : 위 문법 규칙을 만족시키는 한, 추가적인 의미 규칙은 없습니다.

비정상 결과: 만일 문법 형식에 위배되면(인자가 존재하면) 문법에 맞지 않는다는 오류 메세지와 대표 명령어를 출력한 뒤, 주 프롬프트를 다시 출력합니다.

```
AccountNumber > manage 1122334455 ↲
. ! ! 오류: 인자가 없어야 합니다.

AccountNumber >
```

정상 결과 1 : 만일 인자가 없으면 사용자 계정의 계좌 목록과 메뉴를 출력합니다.

```
AccountNumber > manage ↲
========================계좌 목록 출력========================
계좌번호                          잔고
383902                  500,000,000,000
123412                  123,456
321432                  3,000


========================계좌 메뉴 출력=========================
1. 계좌 선택              2. 계좌 생성(공용 계좌)          3. 권한 요청
AccountNumber >
```

7.7.1 부 프롬프트 1: 계좌 선택

문법 형식: 이 명령은 인자가 단 하나여야 하고 숫자여야 합니다.

인자 의미 규칙: 사용자 계좌 목록에 존재하는 계좌 번호 여야 합니다.

비정상 결과: 만일 문법 형식이나 의미 규칙에 위배되면 그에 상응하는 오류 메세지를 출력한 뒤, 사용자 입력을 다시 받습니다.

정상 결과 : 계좌가 선택되고 주 프롬프트로 넘어갑니다.

7.7.2 부프롬프트 2: 계좌 생성(공용 계좌)

문법 형식 : 이 명령은 인자를 허용하지 않으며, 반드시 명령어 단독으로만 사용해야 합니다.

인자 의미 규칙 : 위 문법 규칙을 만족시키는 한, 추가적인 의미 규칙은 없습니다.

비정상 결과 : 만일 문법 형식에 위배되면(인자가 존재하면) 문법에 맞지 않는다는 오류 메세지를 출력하고 사용자 입력을 다시 받습니다.

정상 결과 : 계좌가 생성되고 계좌 목록과 계좌 메뉴가 출력됩니다.

7.7.3 부 프롬프트 3: 권한 요청

현재 사용자의 권한이 부여되지 않은 계좌에 권한을 요청합니다.

문법 형식 : 이 명령은 인자가 단 하나여야 하고 6자리 숫자여야 합니다.

인자 의미 규칙 : 위 문법 규칙을 만족시키는 한, 추가적인 의미 규칙은 없습니다.

비정상 결과 : 만일 문법 형식에 위배되거나 입력한 계좌번호가 존재하지 않거나 이미 해당 계좌에 권한이 있을 시 오류 메세지를 출력한 뒤 사용자 입력을 다시 받습니다.

정상 결과 : 아직 권한이 없는 다른 계좌에 권한을 요청합니다.