

과 목 명 : 알고리즘개론

담당교수 : 조대호 교수님

과 제 명 : 기말 프로젝트

4조

2015313254 노인호

2018311320 권우진

2016311072 신정환

제 출 일 : 2019. 12. 11

**1. Member Role**

노인호 : (1) 예약 레코드 구현 및 테스트 : record.c, record.h recordtest.c 작성

(2) 메인 디자인 및 여러가지 메뉴 선택 기능 구현 및 테스트 : UImain.c, UI.c, UI.h 작성

(3) 전체 프로그램 구조 설계

권우진 : (1) 모듈 구조 디자인, table 모듈 구현 및 테스트 : table.c table.h table\_test.c 작성

(2) 최소 기능 이외 좌석 및 요금 관련 연산 처리 기능 추가 작성

신정환 : (1) Graph, Path 모듈 구현 및 테스트 : Graph.c, Graph.h Path.c Path.h PathTest.c 작성

(2) 도시, 경로 생성 및 최단 경로 검색 기능 구현

**2. Introduction (description of the system objective, available functionalities, and restrictions)**

우리의 목표는 현실 세계의 실제 비행 예약 시스템을 모티브로 하여 보다 작고 제한된 환경에서의 비행 예약 시스템을 구현하는 것이다. 시스템 환경 내에는 26개의 도시와 100개의 직접 경로 비행편(특정 두 도시간의 이동)이 존재하며 기간은 특정 month의 1일 ~ 31일로 제한된다. 직접 경로의 출발지와 도착지는 고정되어 있고 매일마다 각 직접 경로의 출발 시각은 달라진다.

이용자는 출발지, 도착지, 출발 날짜를 입력하여 예약을 할 수 있고 예약을 하게 되면 각 예약의 예약 번호가 생성되어 이용자는 예약 번호를 통해 예약 확인과 예약 삭제가 가능하다. 각 예약에서 이용자가 거쳐가는 경로는 100개의 직접 경로에 대하여 비행 거리를 기준으로 최단 거리를 택하고, 각 경로의 날짜 별 출발 시각을 기준으로 최단 대기시간(가장 빨리 탈 수 있는)을 택하는 경로이다. 이용자는 예약을 하게 되면 도착 날짜와 시간까지 바로 알 수 있다.

실제 환경과 비교하여 한계점을 생각해본다면 본 시스템에서는 매일 출발 시각이 달라지기는 하지만 100개의 직접 경로가 고정되어 있다. 하지만 현실에서는 비행 경로가 고정되어 있지 않고 계속 바뀐다. 또한 예약을 할 때 항상 최단 경로와 최단 대기시간을 기준으로 비행편이 자동으로 선택되지만 하지만 현실 세계의 이용자는 추가적인 고려사항(휴식, 가격 등)으로 인해 최단 경로로 가지 않을 수도 있고 경유지에서 오래 머무를 수도 있으므로 현실 세계의 시스템과의 차이가 발생한다.

또한 프로그램 자체의 한계점은 이 예약 시스템이 윈도우 시스템에서만 돌아간다는 것이다. 키보드 입력 함수를 conio.h 헤더파일에 있는 kbhit()함수를 사용하였는데, 이 헤더파일은 리눅스에서는 사용할 수 없고, 따로 함수를 구현해야한다. 그리고 화면을 클리어하는 시스템 콜 함수를 사용하였는데 윈도우와 리눅스가 system(“cls”)와 system(“clear”)로 서로 다르므로 다른 운영체제에서는 사용할 수 없다. 또한 인코딩이 utf-8로 되어 있어 출력 인터페이스가 utf-8인코딩을 지원하지 않으면 글자가 정상적으로 나오지 않을 수 있다. 윈도우 콘솔에서 사용할 경우 chcp 65001을 입력하여 utf-8인코딩을 활성화 시키고 실행한다. Visual studio를 사용하는 경우 UImain.c, UI.c, UI.h, table.c table.h, record.c, record.h, Graph.c, Graph.h, Path.c, Path.h 를 한 프로젝트 안에 넣고 컴파일 하면 exe파일이 생성된다. Gcc를 사용하는 경우 다음 명령어를 입력하여 컴파일 한다. Team4.exe파일이 생성된다.

gcc -o team4 UImain.c UI.c UI.h Path.c Path.h Graph.c Graph.h table.c table.h record.c record.h

**3. User interface and modules description (including inputs and outputs)**

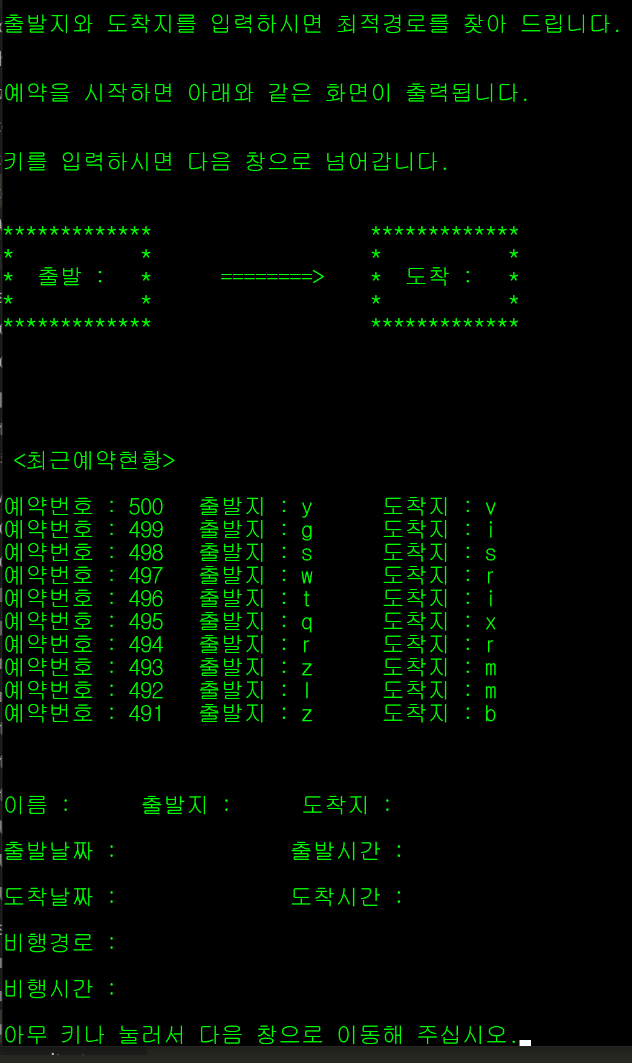
Graph

Path

Table

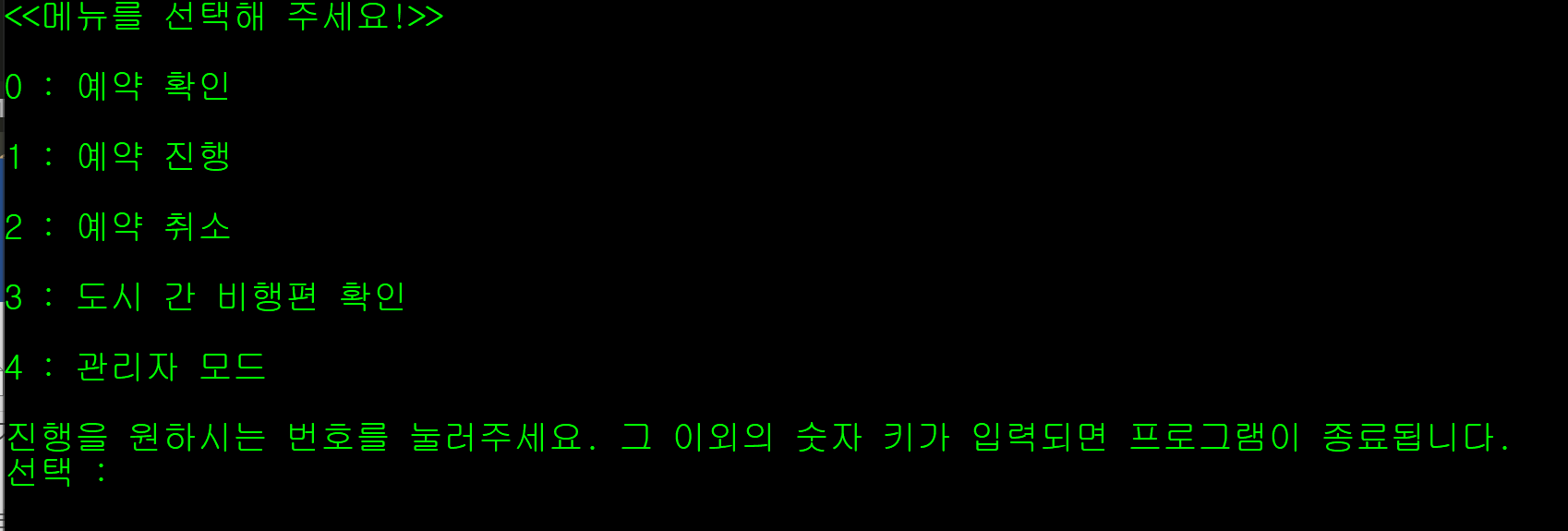
UI

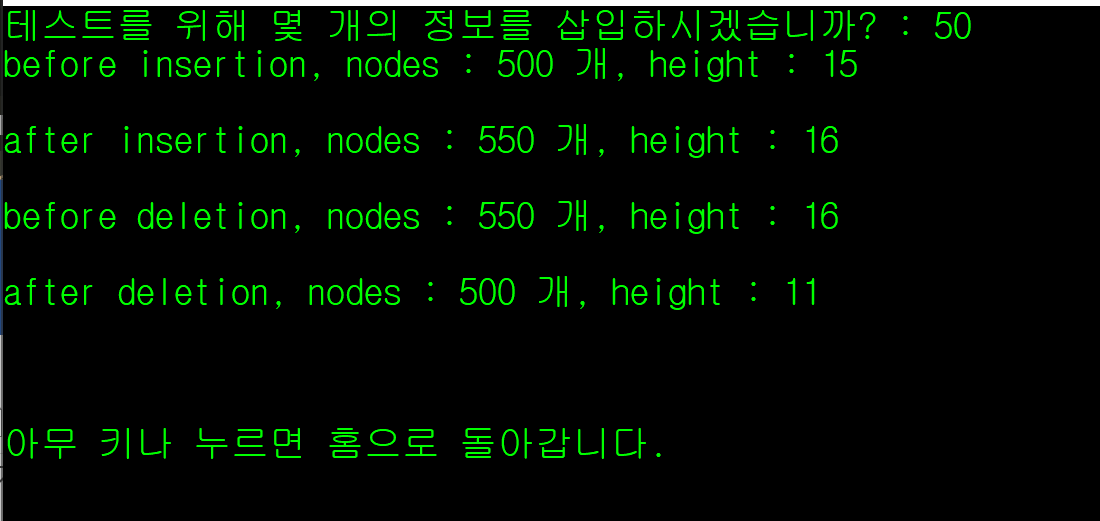
Main

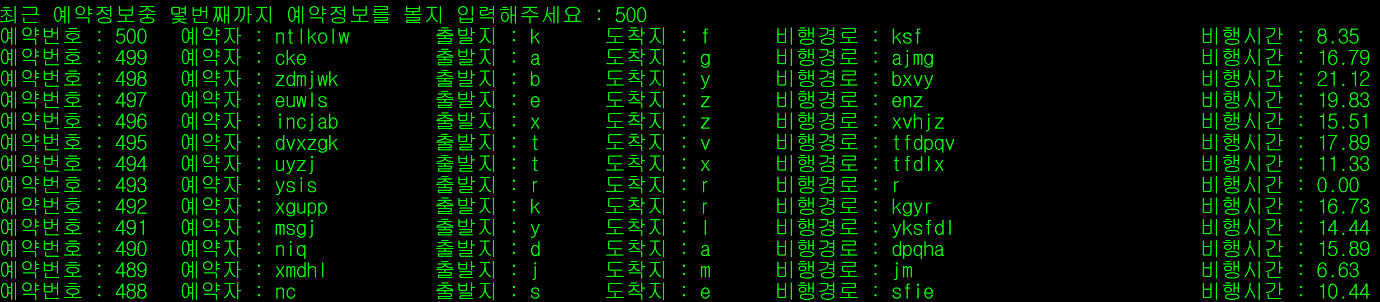
모듈성을 최대화하기 위해 위와 같은 3-레벨 계층의 모듈의 활용이 가능한 환경을 구현하였다.

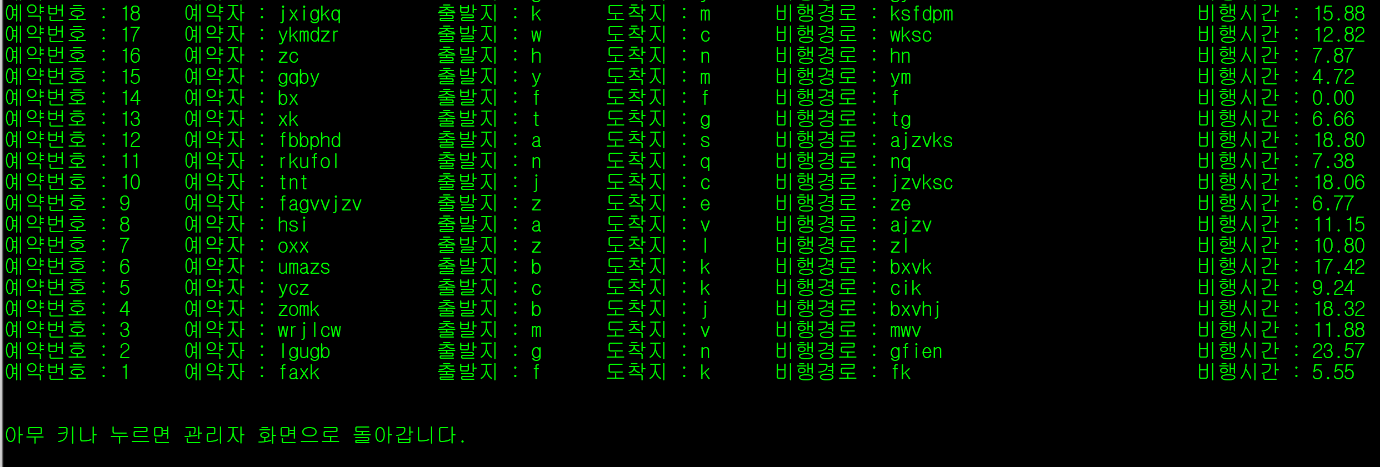
(1) 유저 인터페이스

먼저 utf-8로 인코딩 했기 때문에 윈도우 콘솔을 쓴다면, chcp 65001을 입력하여 utf-8 인코딩을 활성화시켜준다. 그 다음 exe파일을 실행시키면 맨 처음 화면에는 예약을 진행했을 때 어떻게 화면 레이아웃이 구성되는지 알려준다. 예약을 했을 때 출발지, 도착지가 네모 칸에 삽입되고, 가장 최근 예약된 순서대로 10개가 출력된다. 다른 사람의 예약정보가 나타나 있으므로 개인정보보호를 위해 예약자 이름은 출력되지 않고 번호와 출발지 도착지만 출력된다. 그리고 그 밑에 예약을 진행한 사람의 예약 번호와 예약자 이름, 출발지, 도착지, 출발 날짜와 시간, 도착 날짜와 시간, 비행 경로, 좌석 등급, 총 요금, 비행 시간이 출력된다. 아무 키나 누르면 메뉴 선택 화면으로 넘어간다. 실행 화면은 옆 그림과 같다.

메뉴 선택 화면에서는 총 5가지의 선택지(예약 확인, 예약 진행, 예약 취소, 도시 간 비행 편 확인, 관리자 모드) 가 있다. 권장되는 번호를 누르지 않으면 프로그램이 종료되게 만들었다. 각 경우에 대한 기능들은 4번에서 자세하게 설명하도록 하겠다.

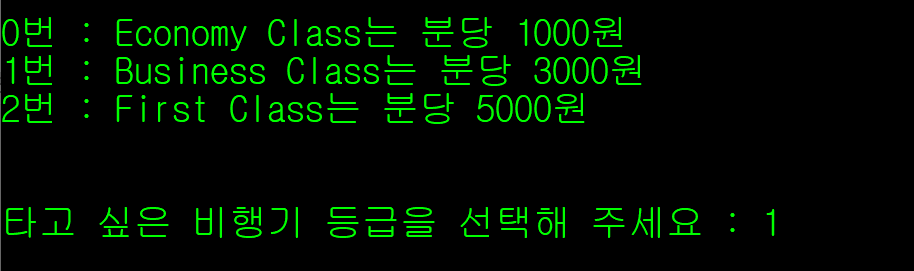
테스트는 관리자 모드에서 모두 진행하도록 만들었다. 관리자 모드는 4번을 입력하여 들어갈 수 있는데 비밀번호를 입력해야 하고, 비밀번호는 algorithm으로 설정하였다. 관리자 화면에서 예약과 취소 테스트를 진행할 수 있고 테스트를 진행하면 예약하기 전 노드 수, 예약 테이블 레드블랙 트리 높이와 예약 후 노드 수, 트리 높이, 그리고 취소 전 노드 수, 트리 높이, 취소 후 노드 수 트리 높이가 출력된다. 예약자 테이블을 이용하므로 실제 시스템에 영향을 주지 않기 위해 삽입과 삭제를 동일한 수만큼 진행하게 된다. 얼마만큼 넣고 뺄 건지는 입력 값으로 들어간다. 다른 수의 삽입과 삭제 테스트를 진행하려면 일반 메뉴에 1번 예약 진행과 2번 예약 취소 과정으로 테스트할 수 있다.

또 다른 관리자 모드에서의 테스트는 예약자 정보 전체 출력 과정을 진행하였다. 예약자 정보 출력 메뉴로 들어가면 총 몇 개의 정보를 조회할 것인지 입력 값을 받는다. 그리고 출력 값으로 입력 값으로 받은 만큼의 수의 예약번호, 예약자, 출발지, 도착지, 최단비행경로, 비행시간이 모두 출력된다.



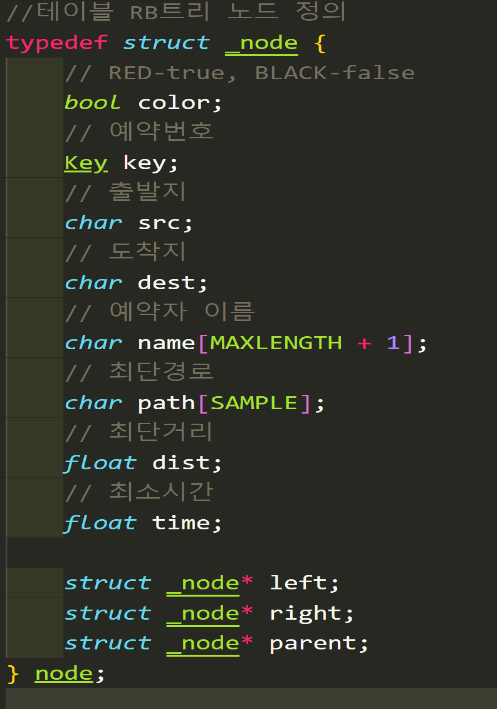
(2) 부가 기능 - 좌석 등급 선택 및 요금 출력

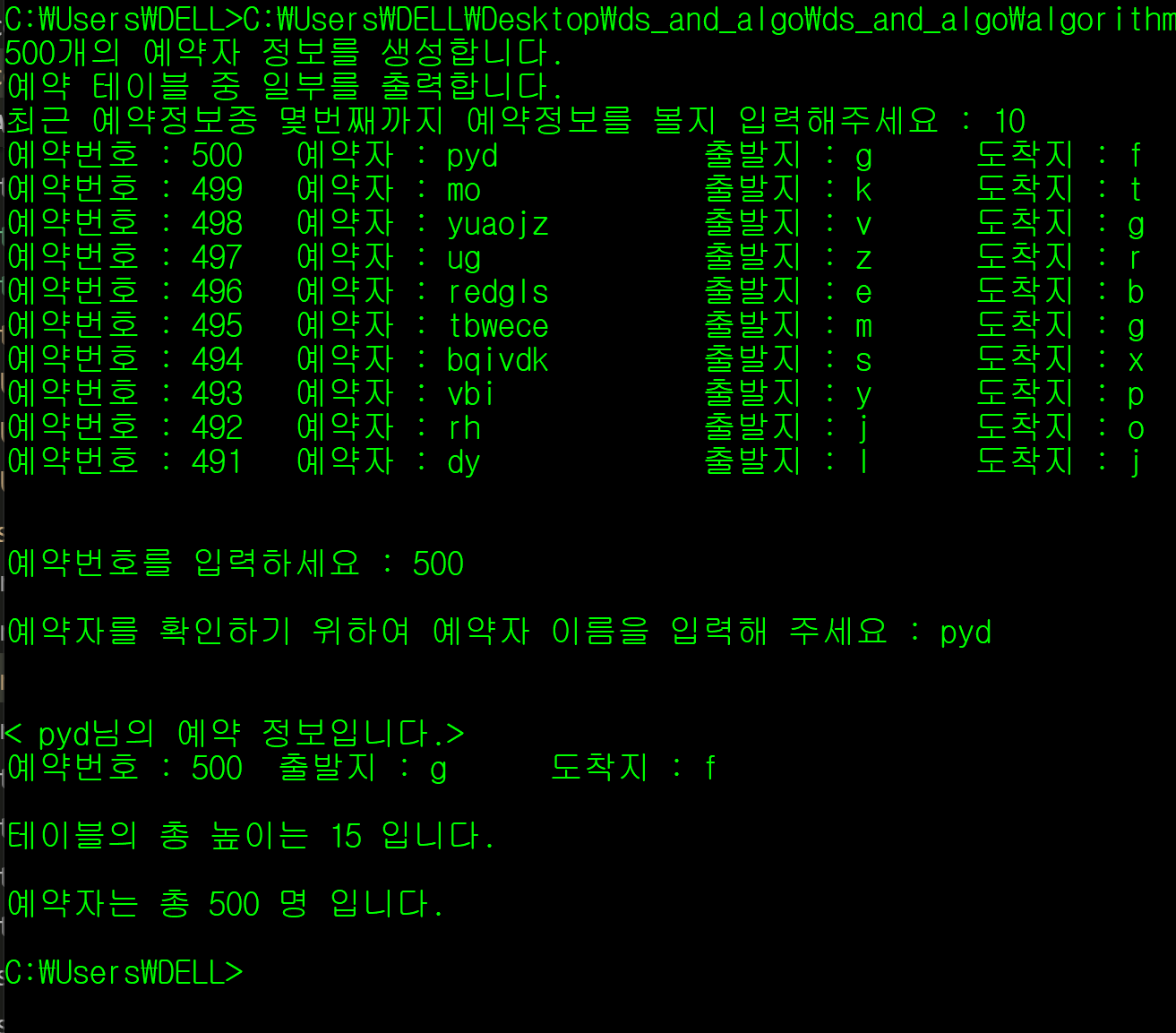
Economy Class, Business Class, First Class중 하나의 등급을 선택하면 정해진 요금 비율에 따라 총 요금이 자동적으로 계산되어 전체 경로를 출력할 때 같이 출력된다. 비행 시간을 기준으로 요금은 각각 분당 1천원, 3천원, 1만원이다. 메뉴에서 1. 예약진행을 누르고 예약을 진행하는 과정에서 볼 수 있다.

입력 예시

출력 예시

(3) record 모듈

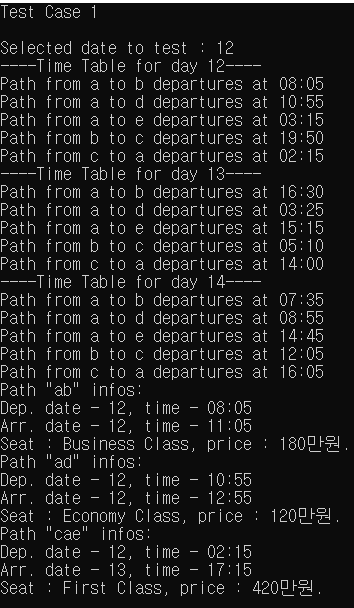
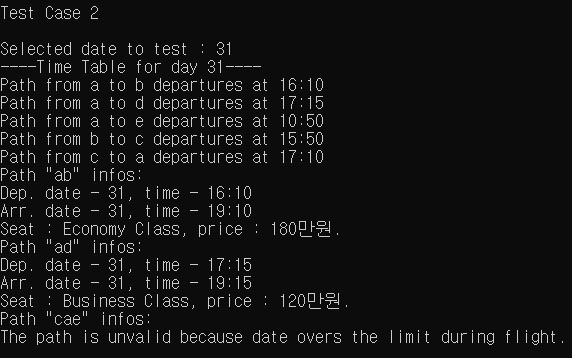
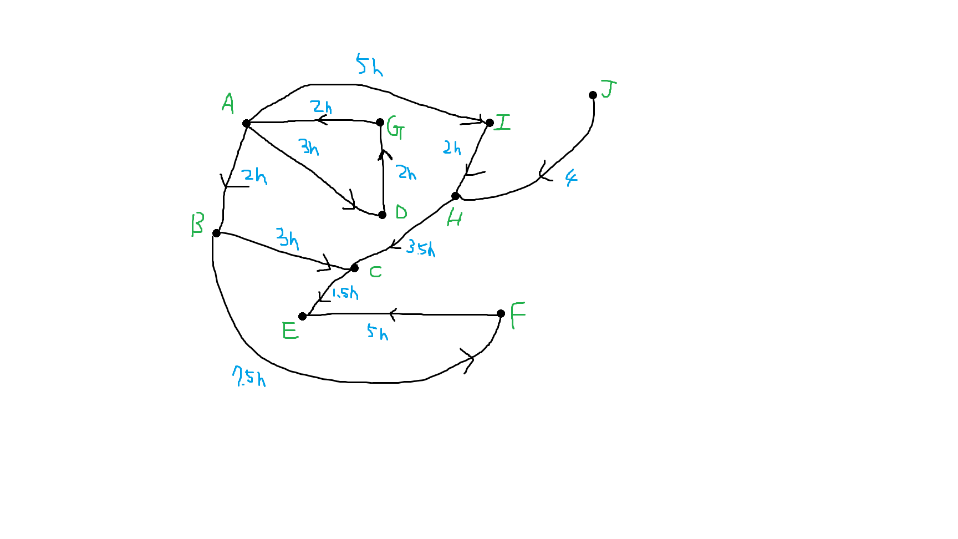
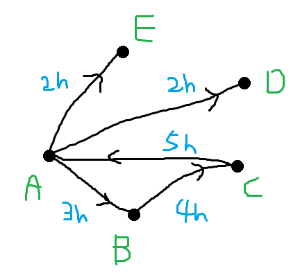
 예약번호를 키 값으로 해서 레드 블랙 트리를 만든다. 레드 블랙 트리의 노드에 포함되는 정보는 옆의 사진과 같다. 이 중 최단 경로와 최단 거리, 최소 시간은 다른 모듈에서 계산되는 값이므로 record 모듈 테스트 시에는 영향을 미치지 않기 위해 레드 블랙트리 삽입함수 RB\_INSERT에는 인수로 포함이 되지 않는다. record.h에는 레드블랙 트리 자료구조와 필요한 각종 변수, 함수들이 선언되어 있고, record.c에는 레드블랙트리 노드 생성, 삽입, 삭제 및 균형과 색깔 변경에 필요한 함수들의 로직이 있다.

recordtest.c에서는 테스트를 진행한다. 먼저 generate함수에서 500개의 예약자 정보를 생성한다. generate함수는 출발지와 도착지를 ‘a’~’z’중 하나로 랜덤 생성하고, 예약자 이름은 1~10의 무작위한 길이를 가진 문자열이면서 각 문자는 ‘a’~’z’중 랜덤하게 결정되고, 중복이 없는 이름으로 생성된다. 그리고 예약자 테이블을 최근 기준으로 몇 번째까지 볼 건지 입력 값을 받으면 그 입력 값만큼 예약 테이블을 출력한다. 이 테이블에는 테스트용 이므로 예약번호, 예약자이름, 출발지, 도착지가 출력된다. 예약번호와 예약자이름을 입력 받고 확인하여 일치하면 예약정보를 출력하고 테이블 높이와 노드 수, 즉 총 예약자 수를 출력한다.

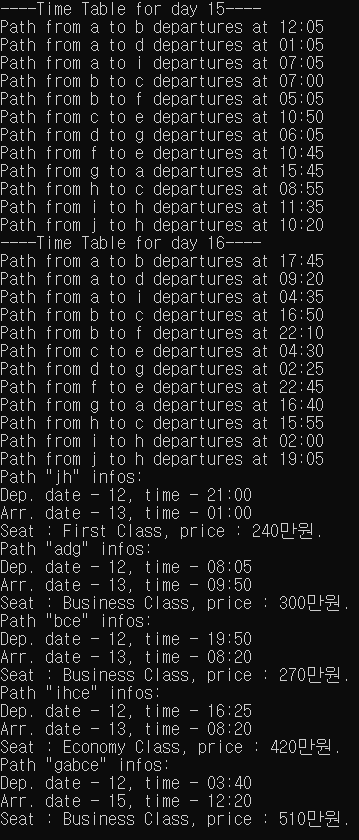
(4) table 모듈 설명

table 모듈은 크게 두 가지 역할을 수행한다. 우선, 각 경로마다 1~31일의 출발 시간을 5분 간격으로 랜덤하게 생성하는 buildTimeTable 함수가 있다. main함수 상에서 그래프의 edge 개수는 100개로 항상 같기 때문에 각 edge에 index값을 주면 2차원 배열로도 충분히 time table을 쉽고 효율적으로 구현할 수 있다. 따라서 time table의 자료구조는 int형의 2차원 배열을 사용하였으며, “시간\*60+분”의 값을 저장하도록 설정하였다. 또한, 위의 path 모듈에서 구해진 최단 경로를 입력 받으면 출발 시간표를 탐색하며 각 비행마다 최대한 빨리 출발할 수 있는 시간대를 구한다. 부가기능의 요금 계산도 여기서 수행한다. 입력 값은 출발 날짜와 좌석 등급이며, 출력 값은 출발 시간, 도착 날짜, 도착 시간, 총 비행 요금이다. 이 기능은 computeFlightInfos 함수가 계산한다.

모듈을 테스트하는 과정은 다음과 같다. 우선, 그림과 같은 두 그래프를 만든다. 그런 뒤, time table이 잘 만들었는지 확인하기 위해 임의의 날짜를 골라 해당하는 날짜의 time table을 출력한다. 각 그래프에 대해 주어진 경로를 설정한 뒤, computeFlightInfos함수를 통해 필요한 정보를 계산한다. 출발 날짜는 앞서 골랐던 날짜로 설정한다. 모든 계산이 완료되면, 이를 출력하여 정상적으로 함수가 작동했는지 확인한다. 이를 테스트한 케이스는 case 1과 3이다.

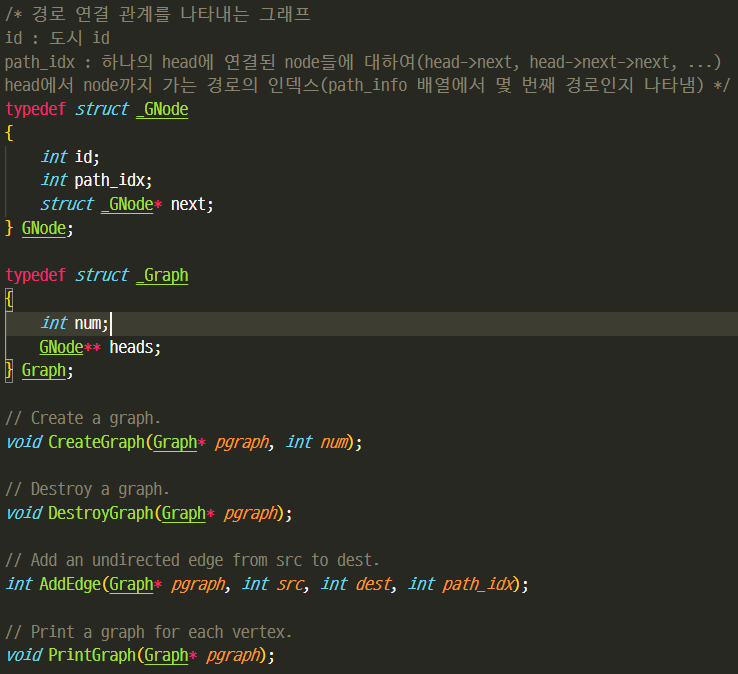
또한, 날짜에 대한 예외 상황을 설정하였다. 주어진 문제에서 별다른 언급이 없었기에, 조원들과 상의하여 합리적인 규칙을 세웠다. 경로 상의 마지막 비행을 하는 도중 32일이 넘어갔으나, 더 이상의 비행 없이 도착지에 도착한 경우, 32일의 time table이 존재하지 않는 것과 별개로 주어진 출발지로부터 도착하는 것이 가능하므로 비행이 가능하다고 판단했다. 이 경우에는 정상적으로 프로그램이 동작한다. 하지만 비행 도중 날짜가 32일로 넘어갔으며, 착륙 후 다시 다른 비행을 해야 하는 경우, 32일의 time table이 없기 때문에 더 이상의 비행을 할 수 없다고 판단하여, 주어진 두 도시간의 비행이 불가능하다고 하기로 약속했다. 이 경우, 주어진 날짜에 출발하는 비행을 끝마칠 수 없다고 출력하였다. 이를 테스트한 케이스가 case 2다.

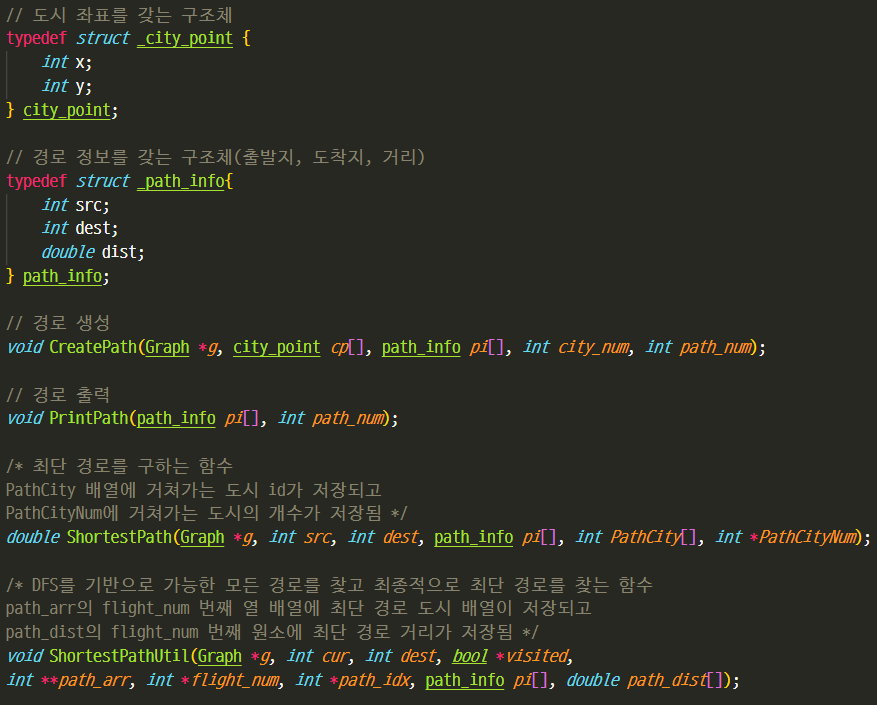
test case 1, 2 test case 3

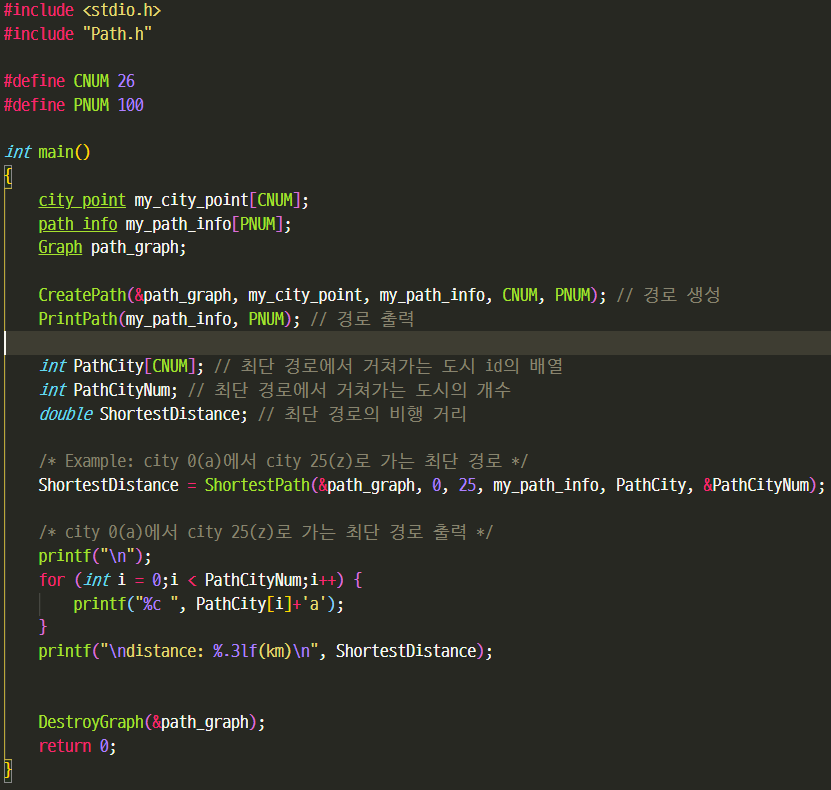
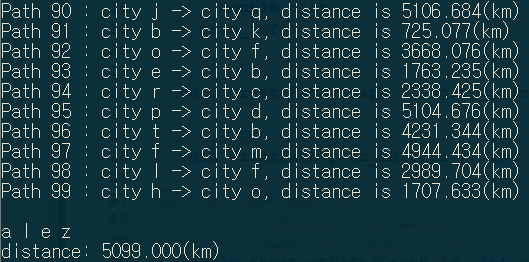


(5) Path 모듈

먼저 Path를 Graph로 구현해야 하므로 자료구조 Graph를 생성하고 Graph를 바탕으로 비행 경로와 관련된 기능을 제공하는 Path를 생성했다. 최종적으로 main에서는 다른 모듈(Reservation record, Time table)과 함께 비행 경로와 관련된 기능을 사용할 수 있다.

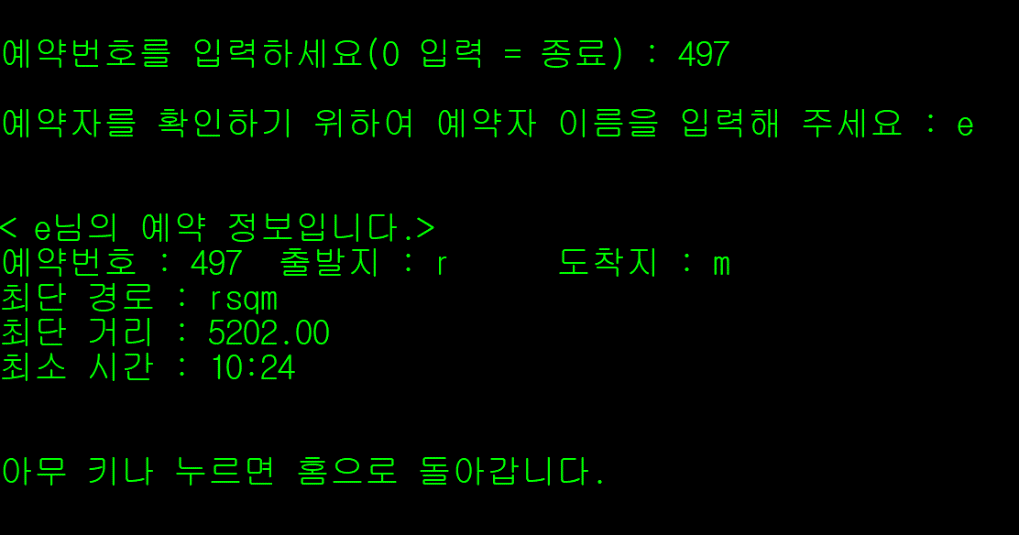
Graph에서 선언된 구조체와 함수이다. Graph를 생성, 삭제하고 edge를 추가하는 Graph 관련 함수와 구조체를 확인할 수 있다.

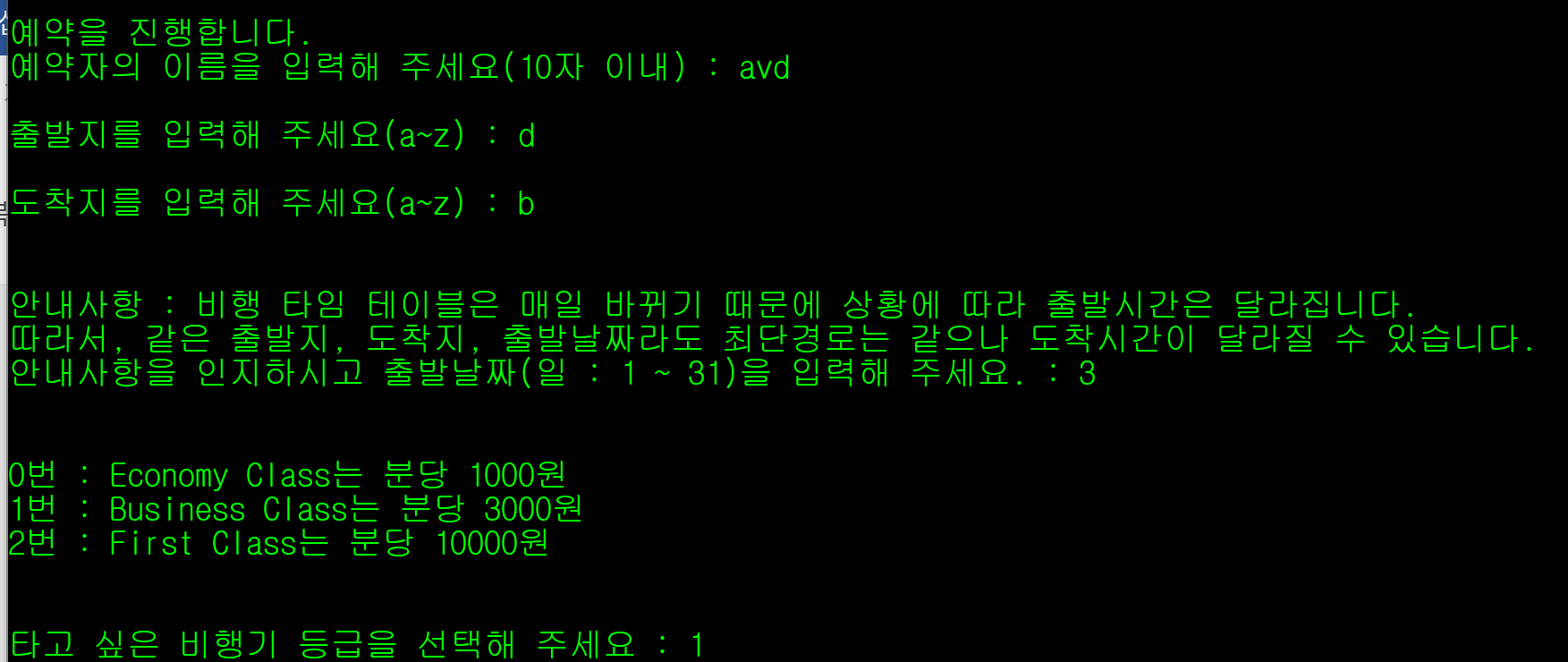
Path에서 선언된 구조체와 함수이다. Path를 생성, 출력하고 최단 경로를 구하기 위한 함수와 구조체를 확인할 수 있다.

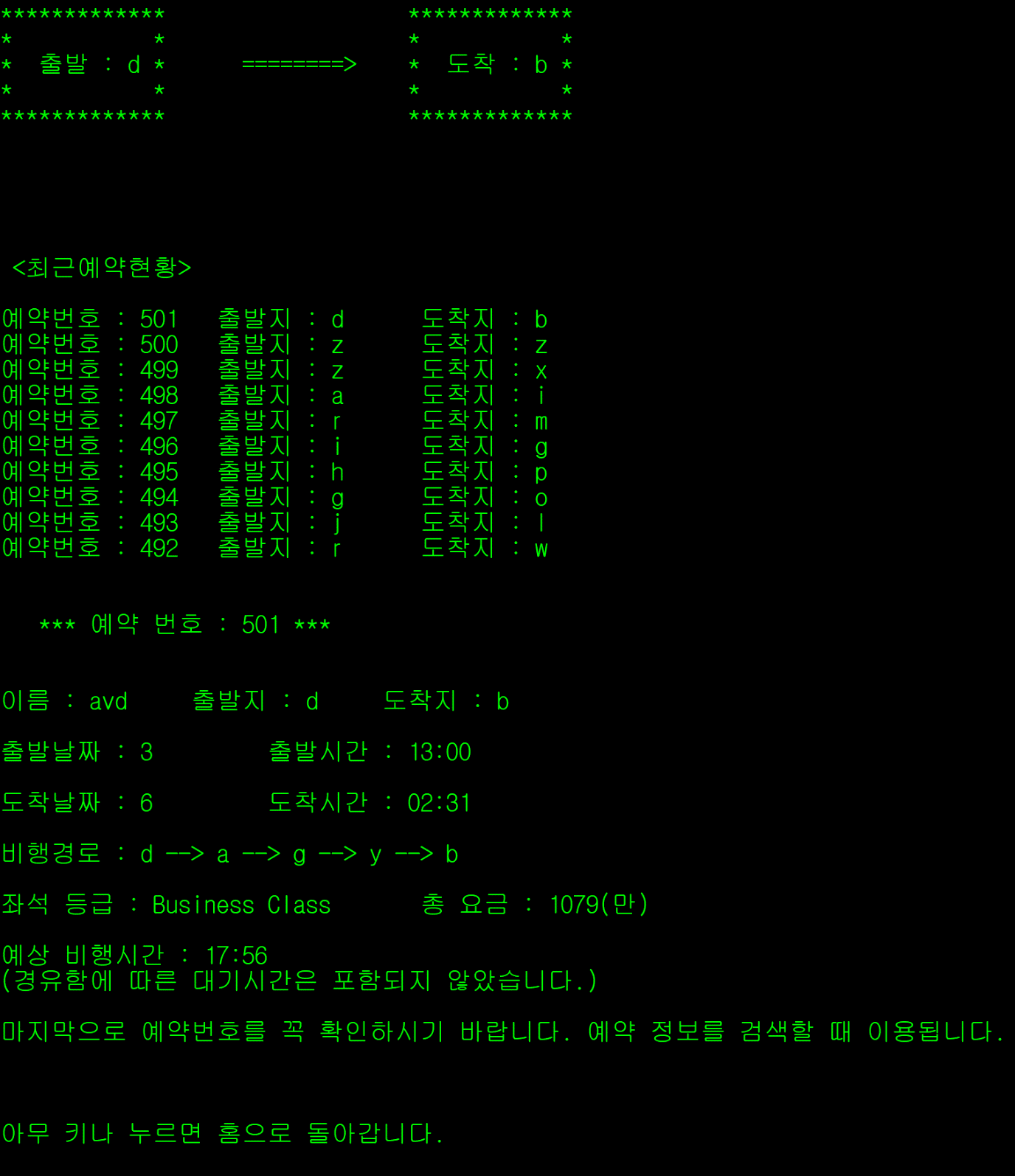
Path 모듈을 테스트하기 위한 Pathtest 프로그램이다. 26개의 도시와 100개의 경로를 생성하고 최단 경로까지 구하는 프로그램의 코드와 실행 결과이다. Path 0~89의 출력결과는 생략하였다.

**4. Extensive execution examples and analysis**

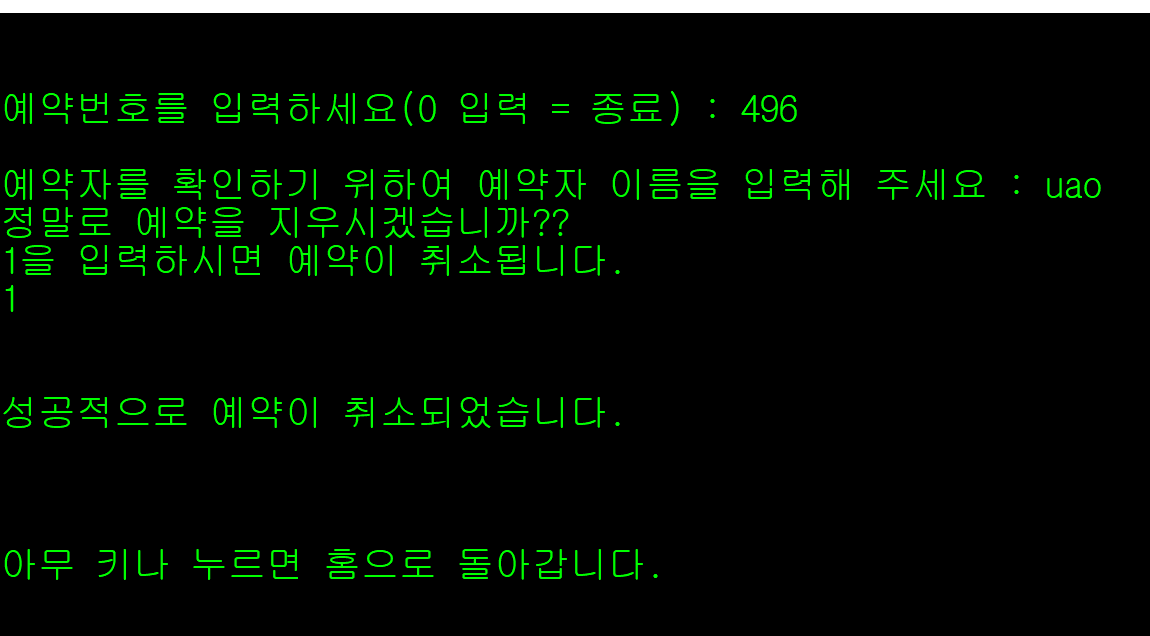
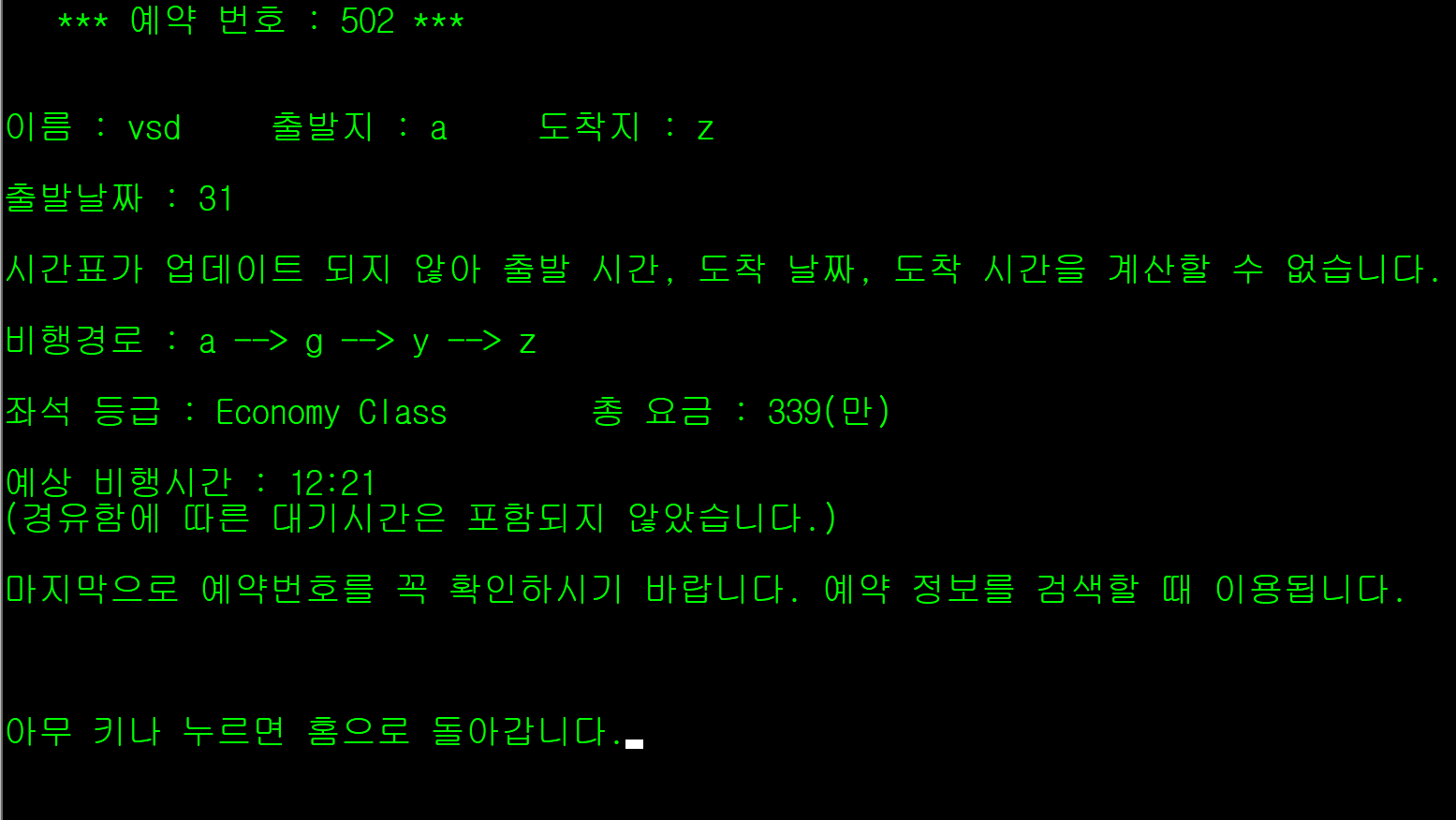
프로그램을 실행하여 메뉴를 확인하는 단계부터 설명을 하고자 한다. 먼저 0번 메뉴 예약확인을 누르면 예약을 확인하는 과정이 시작된다. 예약번호와 예약자 이름을 넣고 일치하면 예약자 정보를 출력하고 일치하지 않으면 다시 번호와 이름을 입력 받는다. 0을 누르면 과정이 종료된다. 출력 결과는 위와 같다.



다음은 1번 메뉴 예약진행 과정이다. 예약자 이름, 출발지, 도착지, 출발날짜, 좌석등급을 선택하면 다음과 같이 결과물이 출력된다. 만약 table 모듈에서 설명한 대로 날짜의 예외사항에 의해 출발시간, 도착날짜, 도착시간이 계산 불가능하다면, 시간을 계산할 수 없다는 안내문이 출력된다.



시간 계산이 불가능한 경우 :

다음은 2번 메뉴인 예약취소 과정이다. 예약번호와 예약자 이름을 입력 받고 일치 여부를 확인한다. 일치할 경우, 예약을 취소시킬지의 여부를 재확인한다. 이 때, 1번을 누르면 취소가 진행되어 레드블랙 트리에서 해당 예약이 삭제된다. 다른 숫자를 입력하면 취소되지 않고 다시 입력 값을 받는다. 다음은 취소를 진행시킨 예시이다.

3번 도시간 비행 편 확인은 2개의 도시간에 어떤 경로들이 있는지 모두 확인하는 메뉴이다. 3번을 누르면 바로 기존에 생성되어 있던 100개의 경로가 모두 출력된다.



4번의 관리자모드는 유저 인터페이스, 즉 UI의 test과정을 구현한 것이다. 자세한 내용은 위의 부분 모듈에 설명되어 있다.

**5. Discussion and conclusion.**

주어진 조건 이외에도 좌석의 등급을 입력하면 요금을 출력해주는 기능을 추가하여 프로젝트를 완성하였다. 조원들이 사용하는 OS 및 IDE가 달라 인코딩이 깨지는 등의 문제가 있었으나, 잘 해결하여 온전한 프로그램을 만드는 데 성공했다. 프로그램 설계 단계부터 구체적으로 구현 방식을 공유하고, 프로젝트를 진행하는 과정에서 조원들끼리 계속 소통하며 효율적으로 진행하였기 때문에 오류 없이 정상적으로 잘 작동하고 기능에 따라 적절하게 모듈화한 프로그램을 만들 수 있었다.