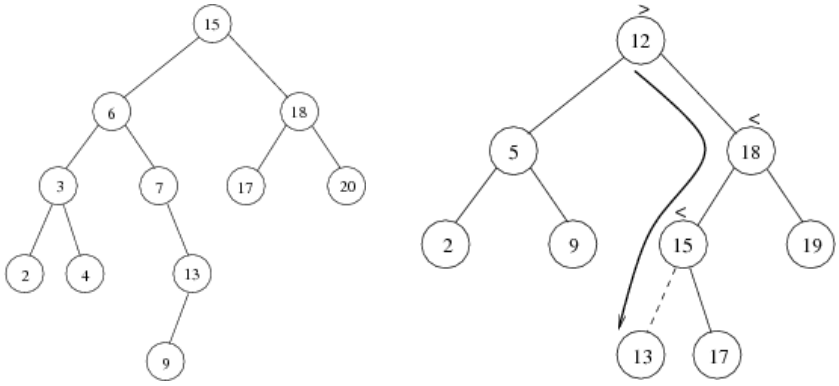
객체지향프로그래밍

3-3

1. [Binary Search Tree] Binary Search Tree는 Figure 1과 같이 1개의 Node가 2개의 Pointer(Child)를 가져 각각의 pointer가 좌측엔 Node보다 작은 값의 Node를, 우측엔 Node보다 큰 값의 Node를 가리키는 구조로 이루어지며 최상단의 Node를 Root, Child가 없는 Node를 Leaf라고 한다. Table 1과 같은 command를 가지고 exit 명령을 받을 때까지 반복해서 입력 받은 command에 대해 동작하는 프로그램을 구현하시오.



<Figure 1. Binary Search Tree 구조(좌) 및 Insert 방법(우)>

Command	사용법	설명
insert	1 integer	binary search tree 에 새로운 integer 를 설치 Figure 1 과 같이 Root 부터 integer 가 node 의 값보다 작으면 좌측, 크면 우측으로 이동하며 탐색해 빈 자리가 있으면 그 자리에 설치한다. integer 가 기존 node 의 number 와 중복될 경우 입력을 무시한다.
Delete	2 integer	binary search tree 에서 integer 를 탐색하고 탐색한 integer Node 를 삭제한다. node 가 삭제되어도 binary search tree 의 구조는 유지되어야 한다. *참고: delete 는 3 가지 경우를 가질 수 있다 1) leaf node 가 삭제될 경우 : leaf node 만 제거 2) child 가 1 개인 node 가 삭제될 경우 : node 를 제거하고 node 의 parent 의 기존 pointer 가 child 를 가리키게 바꾼다. 3) child 가 2 개인 node 가 삭제될 경우 : 제거될 node 의 우측 subtree 중 가장

		작은 node 와 값을 바꾸고 값을 바꾼 node 를 제거한다.
Find	3 integer	integer 를 binary tree 에서 탐색하고 root 부터 해당 node 까지의 경로를 모두 출력한다.
Exit	4	program 종료

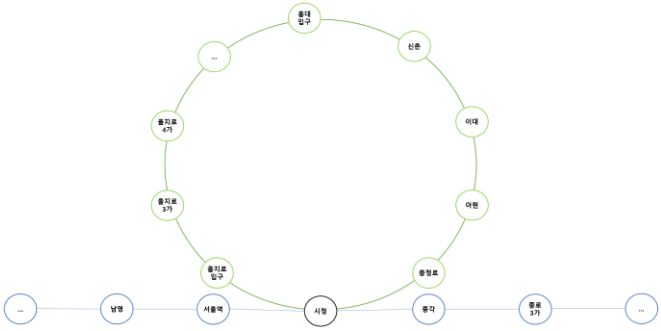
<Table 1>

```
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 15
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 6
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 3
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 2
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 4
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 18
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 7
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 17
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 20
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 13
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 3 9
Path : 15->6->7->13->9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 2 7
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 3 9
Path : 15->6->13->9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 4
```

<실행 예시>

2. [Linked List]지하철의 최단거리를 계산하는 프로그램을 구현하시오. 프로그램은 Assignment3-3-2-1.txt에서 1호선 데이터를, Assignment3-3-2-2.txt에서 2호선 데이터를 읽어와 2호선은 모든 Node가 양방향으로 연결되어 있고 Head와 Tail이 연결된 Doubly Circular Linked List로 구성되어 있으며 1호선은 Doubly Linked List로 연결 되어있다. 프로그램은 예시와 같이 사용자로부터 출발역과 도착역을 입력 받으며 가장 적은 역을 거쳐 도착할 수 있는 경로를 출력한다.

1호선과 2호선은 시청역에서만 환승할 수 있다고 가정하며 1호선 데이터와 2호선 데이터는 과제와 함께 주어진 텍스트파일을 사용한다.(텍스트 파일은 수정하지 않으며 Figure 1 외에 다른 구조에 역 정보를 저장하지 않는다.)



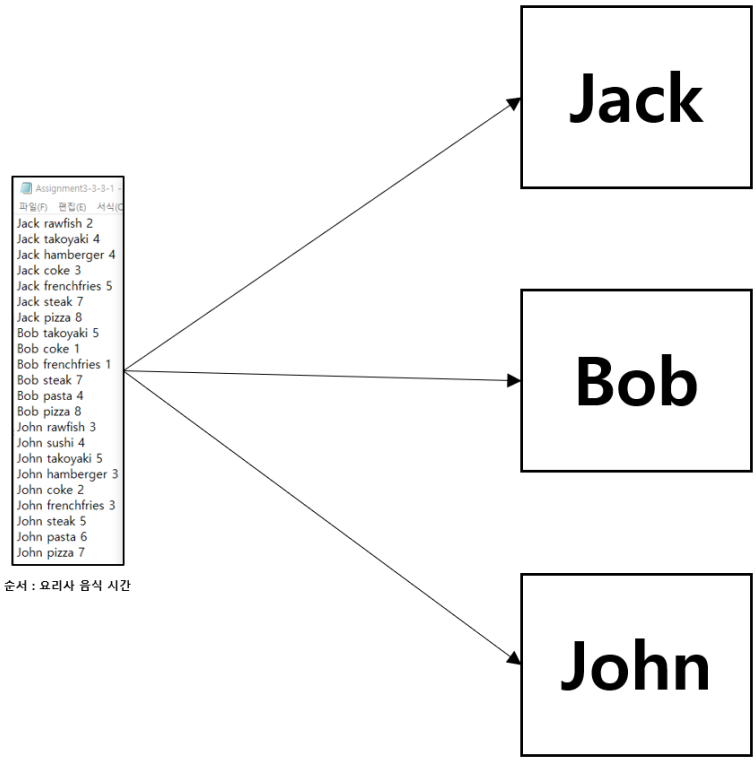
<Figure 1. 프로그램 구조 예시>

```
출발역 : 녹천
도착역 : 이대
녹천->월계->광운대->석계->신이문->외대앞->회기->청량리->제기동->신설동->동묘앞->동대문->종로5가->종로3가->종각->시청->충청로
->아현->이대
```

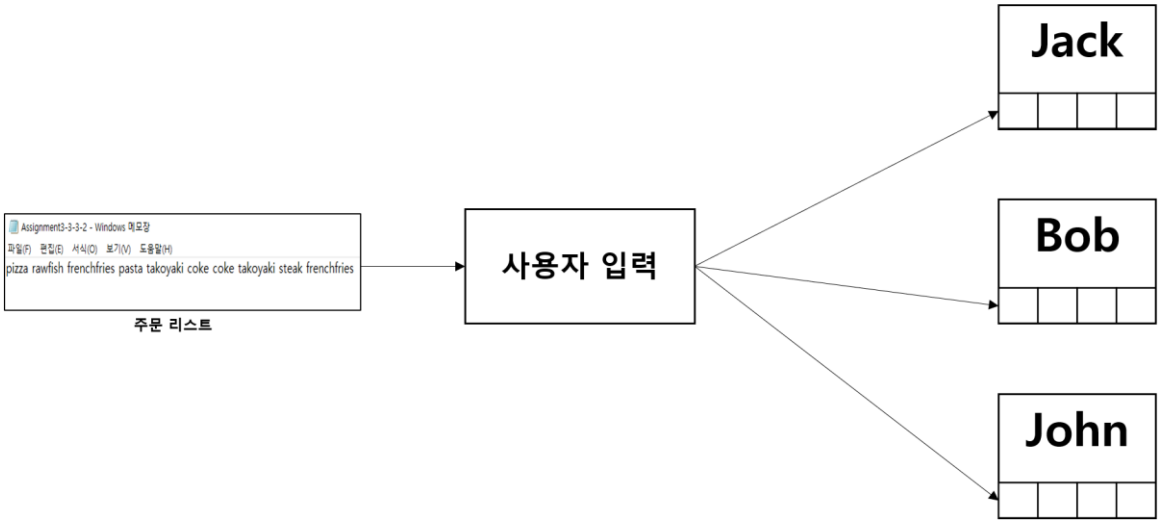
<실행 예시>

3. [Queue, Linked List] 정해진 주문을 요리사에게 할당해 요리하는 시뮬레이터 프로그램을 구현하시오. 프로그램은 시작 시 제공받은 Assignment3-3-3-1.txt를 읽어 3명의 요리사(Jack, Bob, John)가 각 요리에 걸리는 시간정보를 저장하고 Assignment3-3-3-2.txt로부터 주문을 읽어 주문마다 어떤 요리사가 요리할지 사용자로부터 입력받아 각 요리사의 Queue에 저장한다 (Assignment3-3-3-2.txt는 임의작성, Figure 및 예제 참고). 이 때 입력 받은 요리사가 할 수 없는 요리면 재 입력을 요청하며 3명의 요리사 모두가 할 줄 모르는 요리는 주문이 오지 않는다고 가정한다.

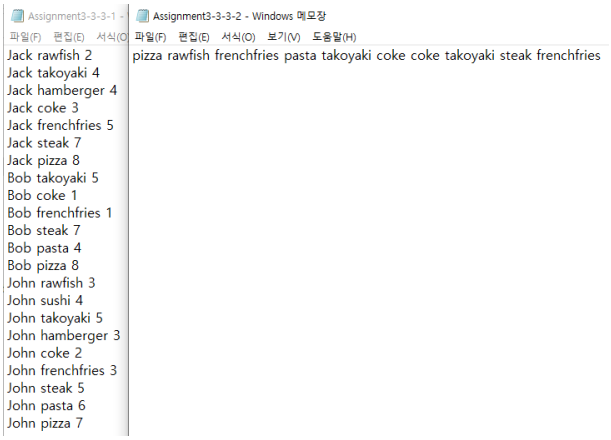
입력이 끝나면 각 요리사는 프로그램 예시와 같이 Queue에 저장된 주문을 요리하며 매 시간마다 어떤 요리를 하고 있는지 현황을 출력한다. 모든 요리사가 요리를 끝내면 얼마나 걸렸는지 출력하고 프로그램은 종료된다.



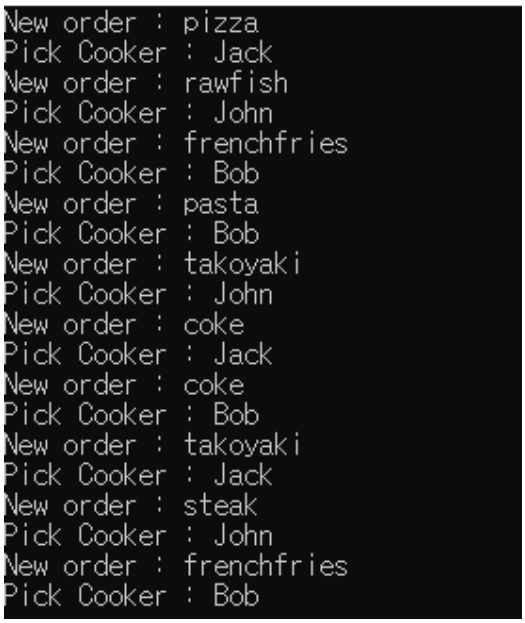
<Figure 1 음식 정보 저장>



<Figure 2 주문을 사용자 입력에 따라 각 요리사의 Queue에 저장>



<Figure 3. 텍스트 파일 예시>



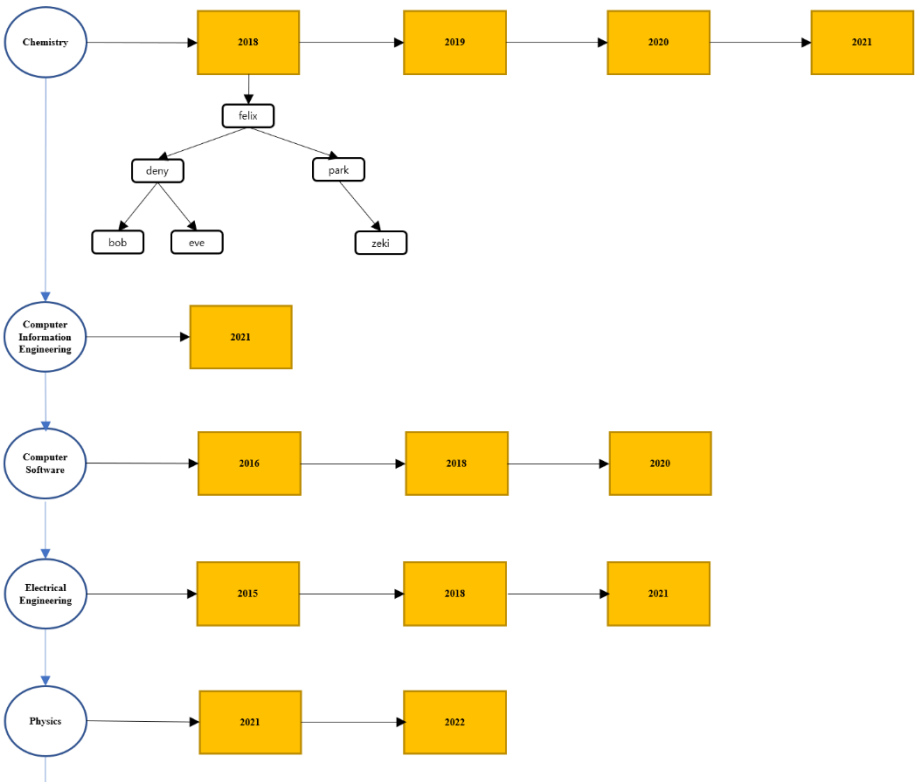
<실행 예시 1 : 시작 시 Assignment3-3-3-2.txt로부터 읽은 주문을 각 요리사에게 할당>

```
Jack
Cooking(pizza7/8)
Cooking(pizza6/8)
Cooking(pizza5/8)
Cooking(pizza4/8)
Cooking(pizza3/8)
Cooking(pizza2/8)
Cooking(pizza1/8)
Done!(pizza)
Cooking(coke2/3)
Cooking(coke1/3)
Done!(coke)
Cooking(takoyaki3/4)
Cooking(takoyaki2/4)
Cooking(takoyaki1/4)
Done!(takoyaki)

John
Cooking(rawfish2/3)
Cooking(rawfish1/3)
Done!(rawfish)
Cooking(takoyaki4/5)
Cooking(takoyaki3/5)
Cooking(takoyaki2/5)
Cooking(takoyaki1/5)
Done!(takoyaki)
Cooking(steak4/5)
Cooking(steak3/5)
Cooking(steak2/5)
Cooking(steak1/5)
Done!(steak)

Bob
Done!(frenchfries)
Cooking(pasta3/4)
Cooking(pasta2/4)
Cooking(pasta1/4)
Done!(pasta)
Done!(coke)
Done!(frenchfries)

Total Time is :15
```

<실행 예시 2 : 각 요리사의 Queue에서 주문을 꺼내 시뮬레이션 출력>

4. [2D Linked List, Binary Search Tree] 학생정보를 관리하는 프로그램을 구현하시오. 프로그램은 Figure 1과 같이 전공, 학번으로 이루어진 Linked List들로 구현되고 학번으로 이루어진 Node마다 이름을 저장하는 Binary Search Tree를 가진 구조로 학생정보를 저장한다. 모든 구조는 숫자는 오름차순, 문자는 대소문자를 구분하지 않고 사전식 정렬을 따른다.

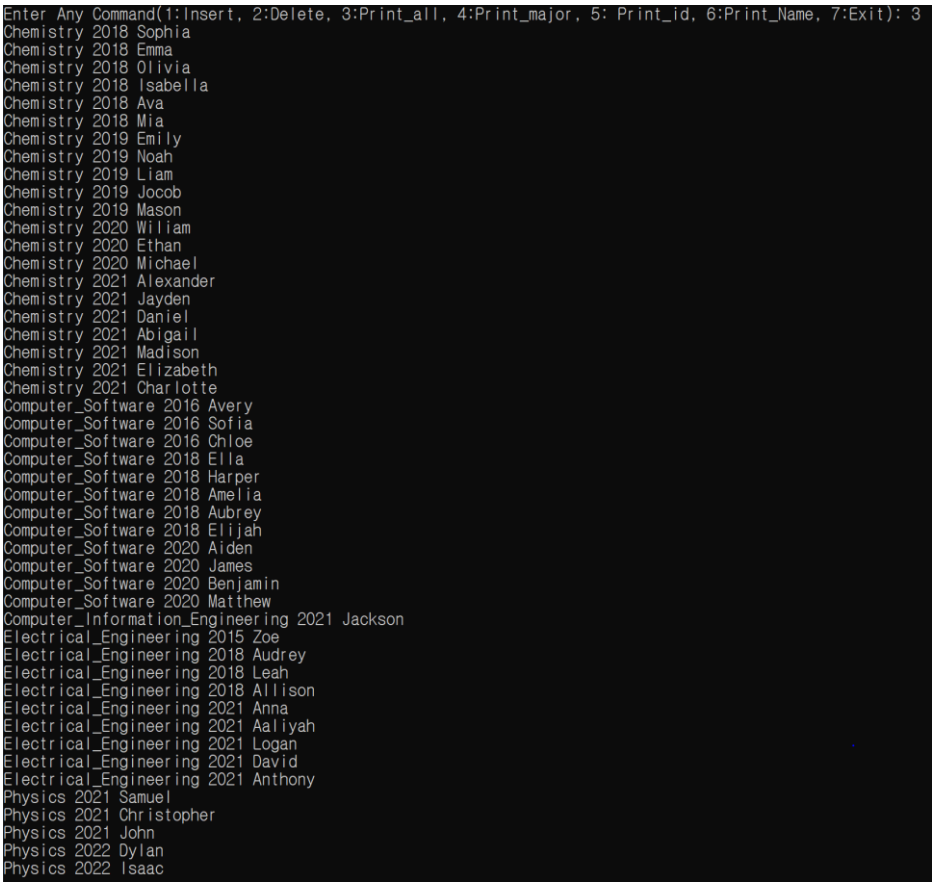
프로그램은 시작 시 Assignment3-3-4.txt로부터 학생정보를 읽어와 자료구조를 구축하고 Table 1에 적혀져 있는 Command에 따라서 동작한다. Assignment3-3-4.txt는 제공된 텍스트 파일로 프로그램은 해당 파일을 읽어 대해서 동작해야 하며 학생 정보는 Figure 1외에 다른 구조로 저장해서 사용할 수 없다.



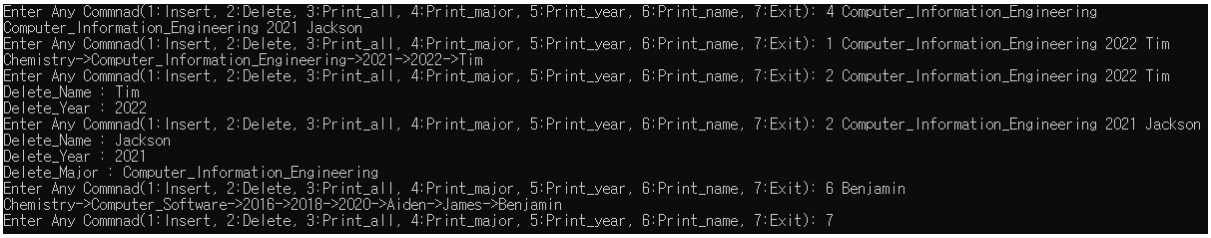
<Figure 1, 모든 학번 노드 아래에 학생이름 정보가 있음>

Command	사용법	설명
insert	1 전공 학번 이름	새로운 학생을 기존의 자료구조의 정확한 위치(Linked List 에서 위치, Tree 내의 위치)에 삽입 삽입 경로 출력(예시 참고), 단 전공/학번/이름이 모두 같은 학생은 없다고 가정
Delete	2 전공 학번 이름	입력받은 정보와 일치하는 학생정보를 제거 제거 후 학번 내에 학생이 없을 경우 학번 Node 제거, 전공 내에 학생이 없을 경우 전공 Node 제거, 제거될 경우 제거 메시지 출력(예시 참고)
Print_all	3	모든 학생 정보 출력(순서 무관)
Print_major	4 전공	전공 내 모든 학생 정보 출력(순서 무관)
Print_year	5 학번	해당 학번의 모든 학생 정보 출력(순서 무관)
Print_Name	6 이름	이름과 같은 학생을 찾아 출력, 탐색 경로를 출력(예시 참고), 이름이 같은 모든 학생의 경로를 출력
Exit	7	Program 종료

<Table 1>



<실행 예시 1 : 프로그램 시작 시 Assignment3-3-4.txt로부터 읽은 학생정보(순서 무관)>


Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 4 Computer_Information_Engineering
Computer_Information_Engineering 2021 Jackson
Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 1 Computer_Information_Engineering 2022 Tim
Chemistry->Computer_Information_Engineering->2021->2022->Tim
Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 2 Computer_Information_Engineering 2022 Tim
Delete_Name : Tim
Delete_Year : 2022
Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 2 Computer_Information_Engineering 2021 Jackson
Delete_Name : Jackson
Delete_Year : 2021
Delete_Major : Computer_Information_Engineering
Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 6 Benjamin
Chemistry->Computer_Software->2016->2018->2020->Aiden->James->Benjamin
Enter Any Commnad(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_year, 6:Print_name, 7:Exit): 7

<실행 예시 2>