PROJECT 1

과목명	데이터구조실습
담당교수	이형근 교수님
학과	컴퓨터정보공학부
학년	2학년
학번	2019202009
이름	서여지
제출일	2020.10.08 (목)

KWANGWOON UNIVERSITY

1. Introduction

이번 과제는 질병 관리 프로그램을 작성하는 것이다. 환자의 이름, 체온, 기침여부, 지역 정보를 파일에서 얻어와 Patient Queue, Location BST와 Patient BST, Location MaxHeap의 세 가지 형태로 정리하는 프로그램이다. 이 프로그램은 command.txt에 적힌 명령어에 따라 data.txt의 정보를 읽고 처리한 뒤, log.txt파일에 결과를 출력한다. 프로그램이 수행할 수 있는 명령어는 모두 7가지이다. 각 명령어의 동작은 다음과 같다.

LOAD	data.txt의 정보를 모두 읽어와 Patient Queue에 저장한다.
ADD	Patient Queue에 환자 정보를 추가한다.
QPOP	Patient Queue에 저장된 정보를 처리하여 Patient BST에 저장한다.
SEARCH	Patient BST에서 특정 환자의 정보를 검색한다.
PRINT	Patient BST/Location MaxHeap의 내용을 모두 출력한다.
BPOP	Patient BST에 저장된 정보를 처리하여 Location MaxHeap에 저장한다.
EXIT	프로그램을 종료한다.

세 가지 자료 형태 중 첫 번째인 Patient Queue는 <queue>라이브러리에서 제공되는 STL을 이용하여 구현한다. data.txt의 자료를 한 줄씩 읽어 ' '문자를 기준으로 이름, 체온, 기침여부, 지역 정보로 나눈 뒤, 해당 정보들을 동적할당한 PatientNode에 넣어 ds_queue에 삽입한다. ds_queue는 BST로 정보를 전달하는 기능에 사용되고, 다른 기능은 하지 않는다.

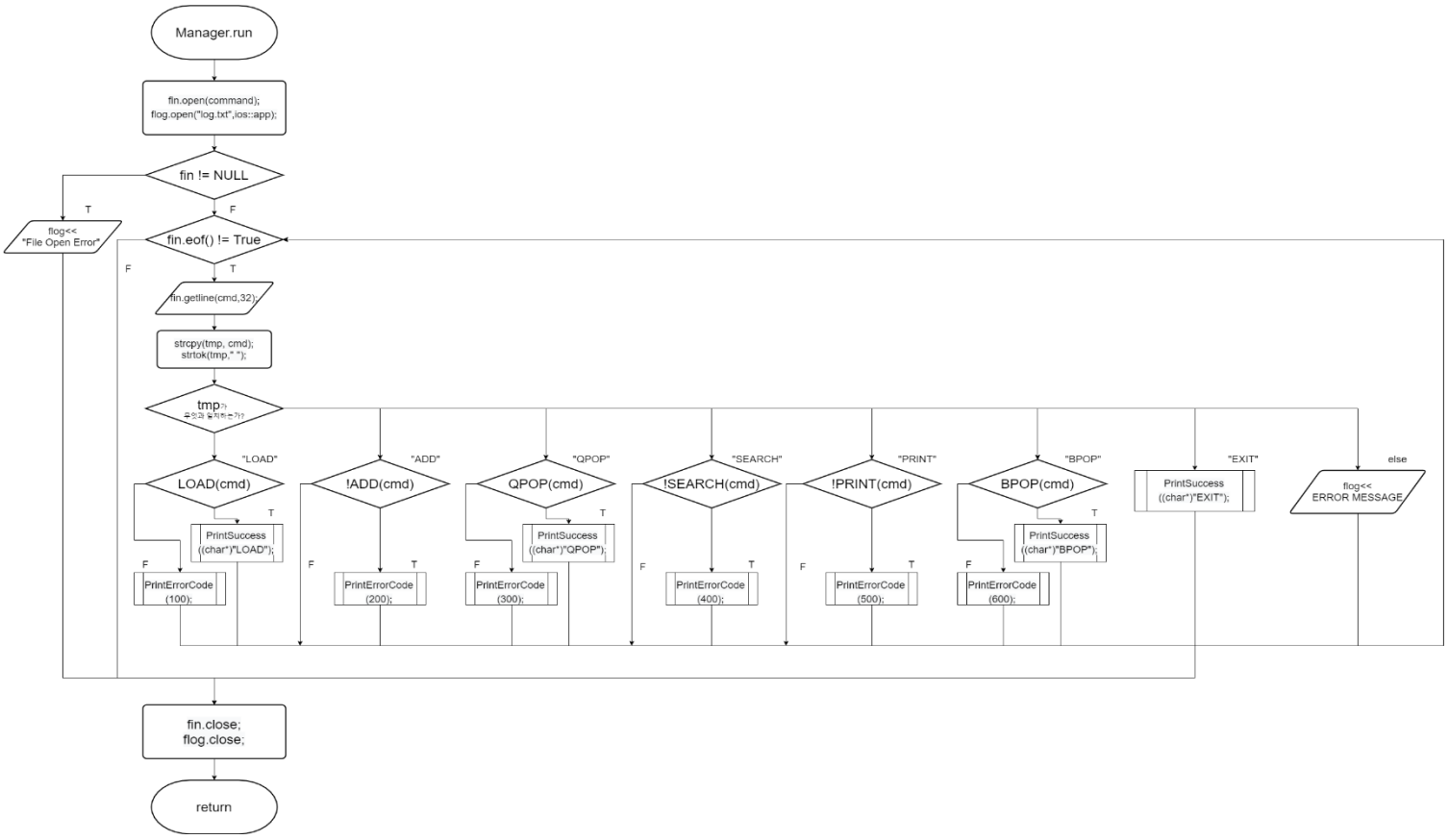
두 번째 자료형인 Location BST와 Patient BST는 각각 클래스를 이용하여 구현하였다. LocationBST는 전체 node의 개수가 정해져있고, 구조가 변경되지 않으므로 Insert_Location함수에서 7개의 LocationNode를 동적할당하고 각각의 지역명을 저장하여 생성한다. LocationBST의 노드는 해당 지역의 PatientBST를 가지고 있다. Patient BST는 Patient Queue에서 정보를 얻어 PatientBSTNode를 생성하여 작성한다. PatientBSTNode는 PatientNode의 환자 이름정보를 그대로 갖고, 체온과 기침여부를 이용해 판단한 질병여부를 갖는다. Patient Queue에서 Patient BST로 정보를 이동시키는 QPOP명령어는 Patient Queue에서 이동시킬 데이터의 수를 인자로 입력받아 동작한다.

마지막 자료형인 Location MaxHeap은 LocationHeap 클래스를 이용하여 구현하였다. LocationHeap의 최대 node수는 7개 이므로 (LocationHeapNode *)를 8개 저장할 수 있는 배열을 동적할당하여 생성한다. Location Max Heap의 경우 Node가 추가되거나 Node에 저장된 정보가 변경되면서 순서가 바뀔 수 있으므로 Insert가 실행되면 Location MaxHeap을 재정렬 한다.

2. Flowchart: 설계한 프로젝트의 플로우차트

(1) 전체 흐름도 - Manager.run()

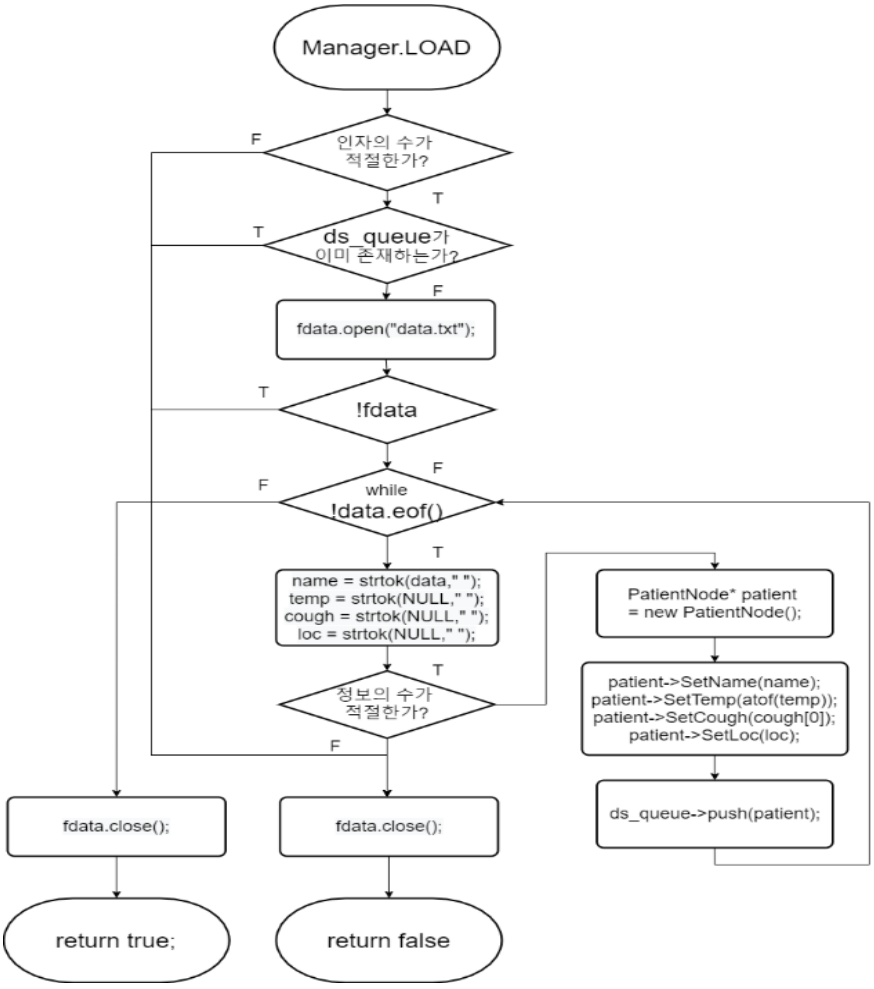
프로그램의 전체 기능을 연결하는 run함수의 순서도이다. command.txt, log.txt파일을 열어 사용하고, command.txt파일의 내용에 따라 각 기능을 하는 함수를 호출한다.



---

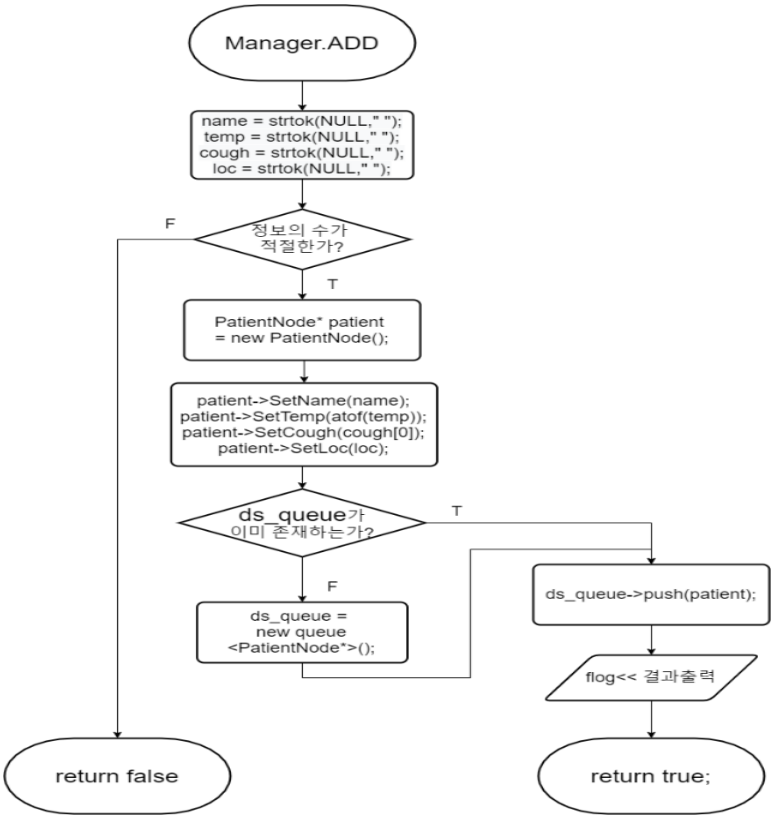
(2) Patient Queue 생성 - Manager.LOAD()

data.txt파일의 정보를 읽어 Patient Queue를 구성하는 함수 LOAD의 순서도이다. 함수가 동작하기 위한 조건을 만족하지 않는 경우 false를 반환하고 종료되고, 모든 조건을 만족한 경우 while반복문 내부에서 PatientNode를 동적할당하며 Queue에 삽입하는 과정을 반복한다.



(3) Patient Node 추가 – Manager.ADD()

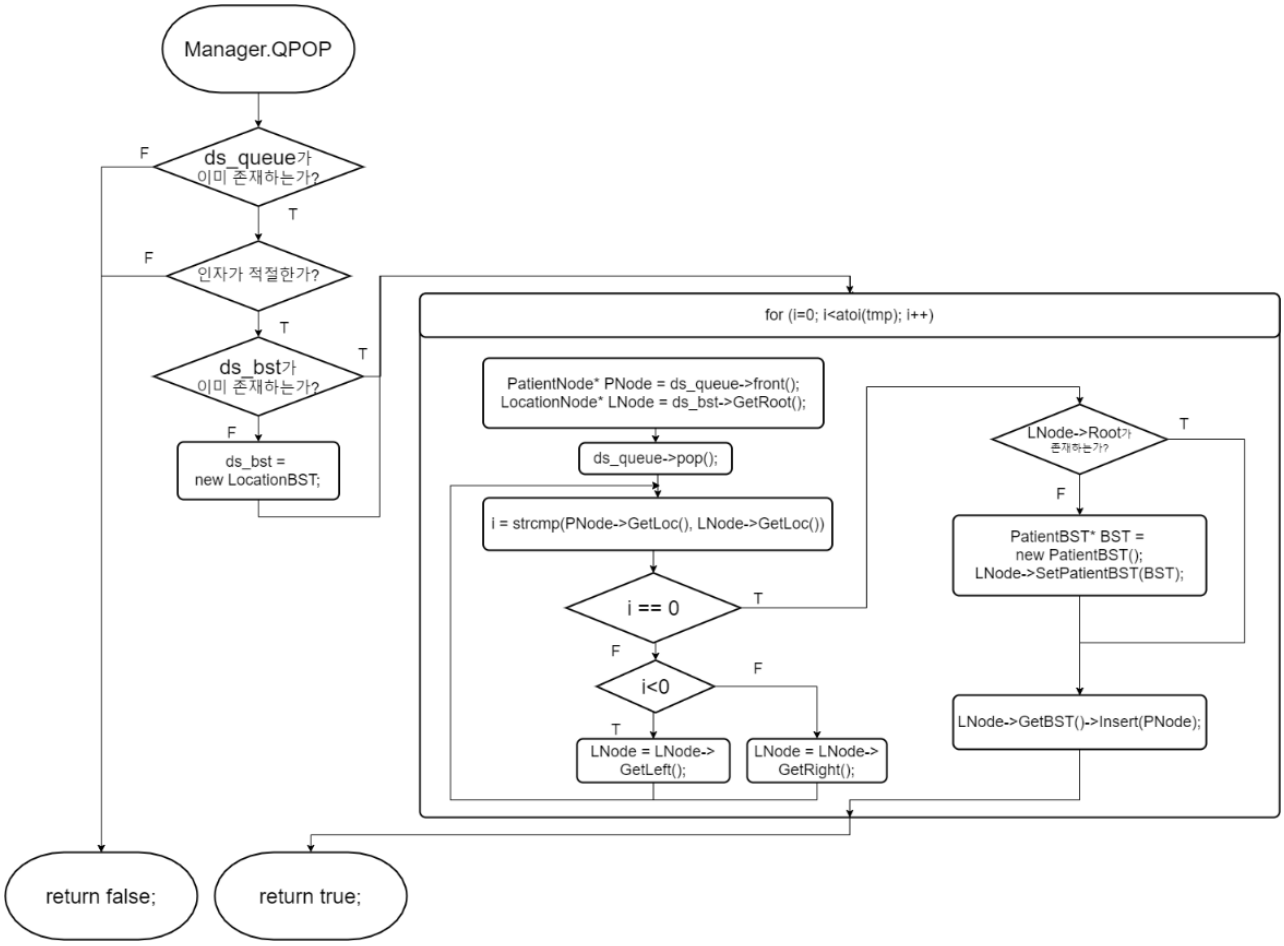
PatientQueue에 새로운 노드를 직접 추가하는 ADD함수이다. 입력받은 정보를 순서에 맞게 ‘ ‘문자를 기준으로 나누고 이름, 체온, 기침여부, 지역이 적절하게 입력된 경우 PatientNode를 동적할당하여 ds_queue에 삽입한다. 이때 ds_queue가 존재하지 않는 경우 새로 생성한다.



(4) Patient BST 생성 - QPOP

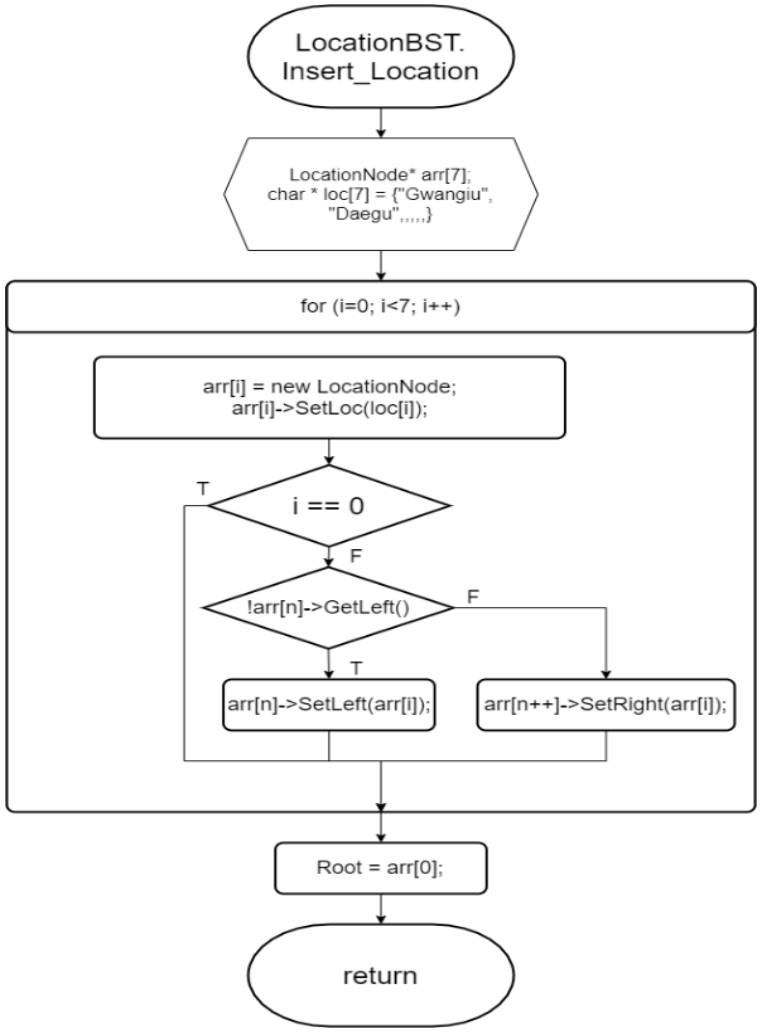
A. Patient Node 생성, LocationNode찾기 Manager.QPOP()

LocationBST와 PatientBST를 생성하고, PatientQueue에서 정보를 받아 해당 지역의 PatientBST에 Node를 삽입하는 기능을 하는 QPOP함수이다. ds_queue에서 정보를 받아 사용하므로 큐가 존재하는지 확인한다. ds_queue가 존재하고, QPOP을 이용해 삽입할 node의 수가 적절하게 입력된 경우 Node를 생성하고 삽입하는 동작을 시작한다. 가장 먼저 ds_bst가 생성되지 않은 경우 LocationBST를 동적할당하여 생성한다. 이후 for 반복문 내부에서 인자로 전달된 수만큼 Node 생성과 삽입을 반복한다. ds_queue에서 dequeue되는 node의 정보를 바탕으로 해당 지역의 LocationBST를 찾고, 하위의 PatientBST에 삽입한다. PatientBST가 존재하지 않는 경우 새 PatientBSTNode를 LocationBST의 Root값으로 저장한다.



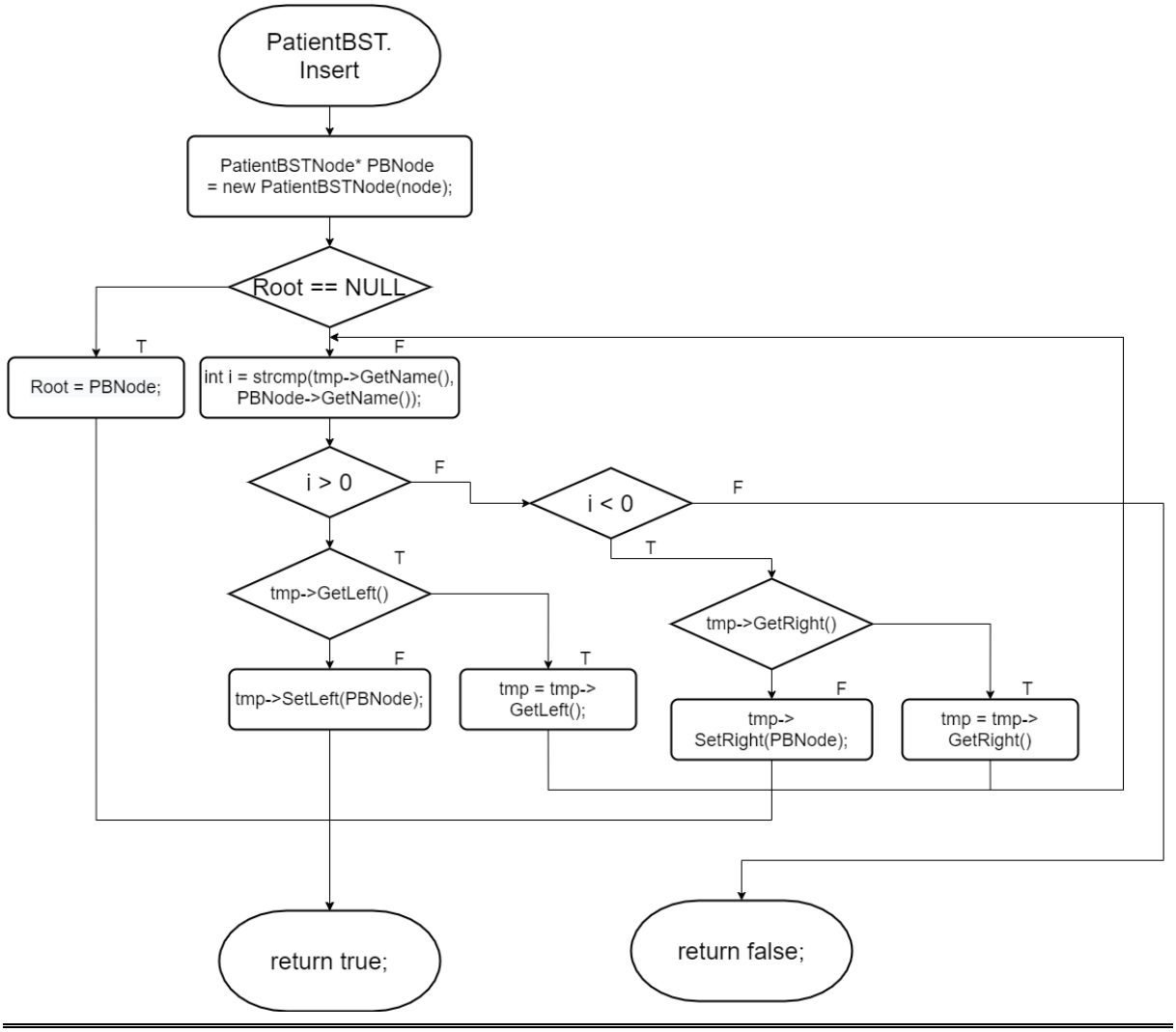
B. Location BST 생성 – LocationBST 생성자

LocationBST를 생성하는 함수이다. LocationBST는 최대 항의 개수가 7개로 고정되어있기 때문에 배열과 반복문을 이용하여 생성하였다. 이 경우 LocationNode의 순서도 변하지 않기 때문에 한 번 생성자에서 생성된 LocationBST는 프로그램이 종료될 때까지 구조가 변경되지 않는다.



C. Patient Node 삽입 – PatientBST.Insert()

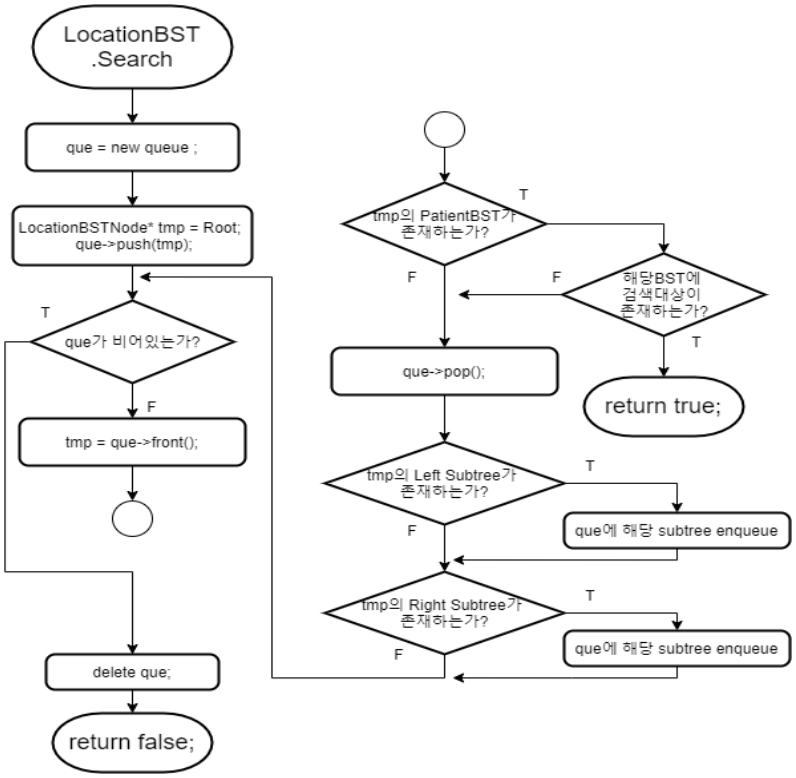
PatientBST에 새로운 node를 삽입하는 Insert함수이다. PatientBSTNode를 동적할당한 뒤, 현재 PatientBST에 Root가 존재하지 않는다면 생성한 PatientBSTNode를 root로 저장하고 종료된다. Root가 존재한다면 각 Node에 저장된 이름 정보를 비교하여 오른쪽/왼쪽 subtree로 이동시켜 저장한다.



(5) Patient BST 검색 - SEARCH

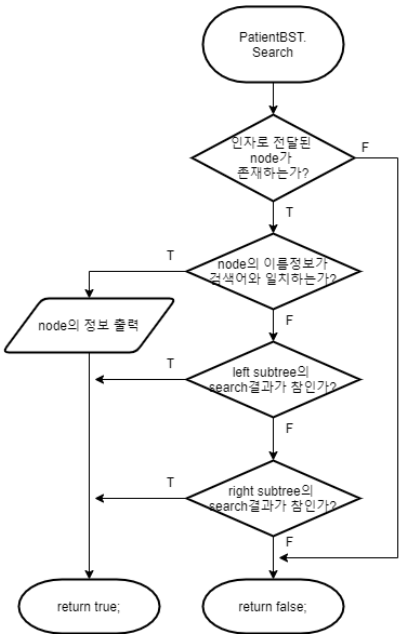
A. LocationBST.Search()

LEVEL traversal을 이용하여 각각 locationBST가 PatientBST를 가지고있다면 해당 bst의 Search함수를 호출한다.



B. PatientBST.Search()

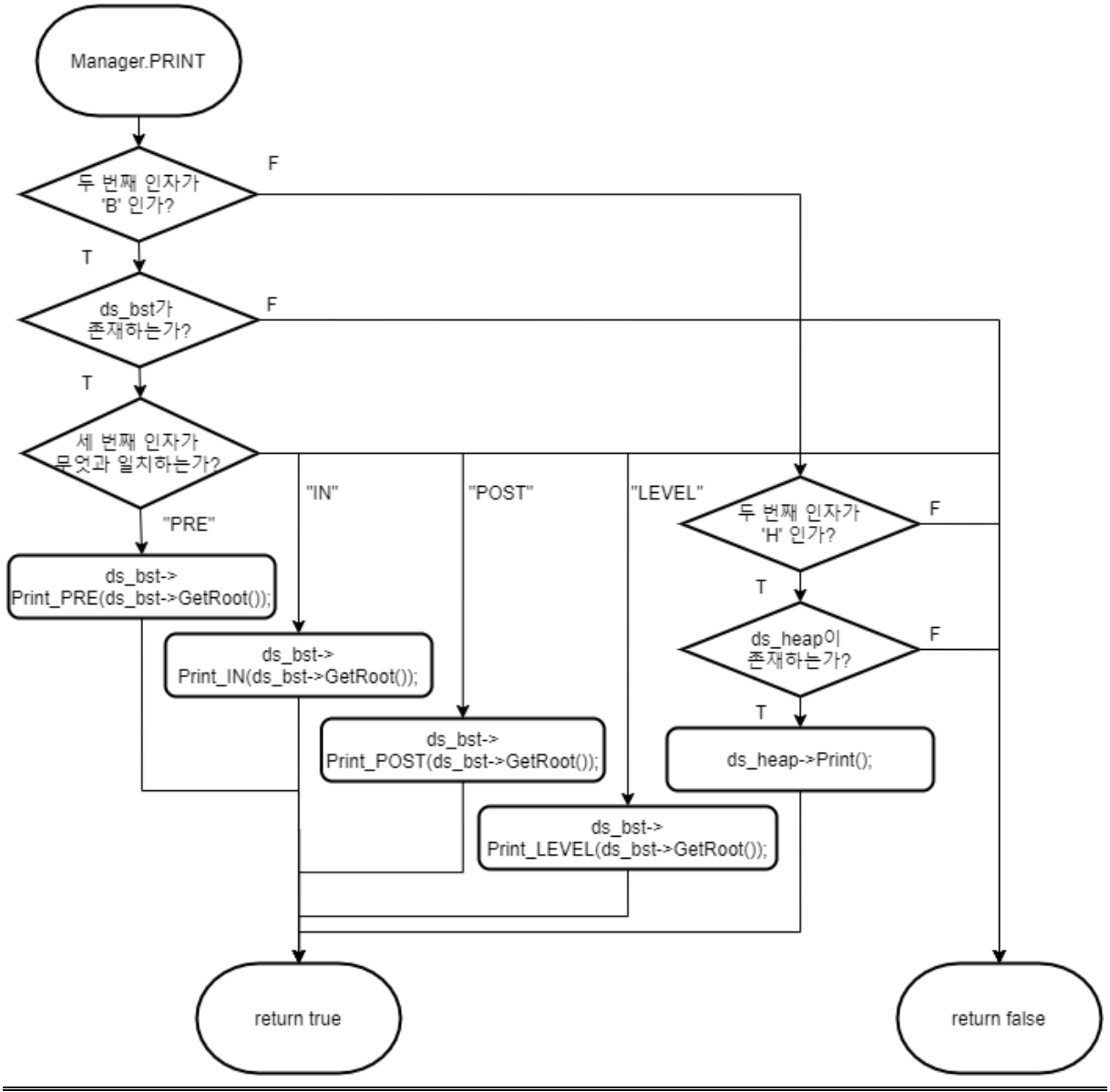
인자로 PatientBSTNode와 검색할 이름을 받아 재귀호출되는 함수이다. PRE order traversal과 유사한 구조를 갖는다.



(6) 출력 – PRINT

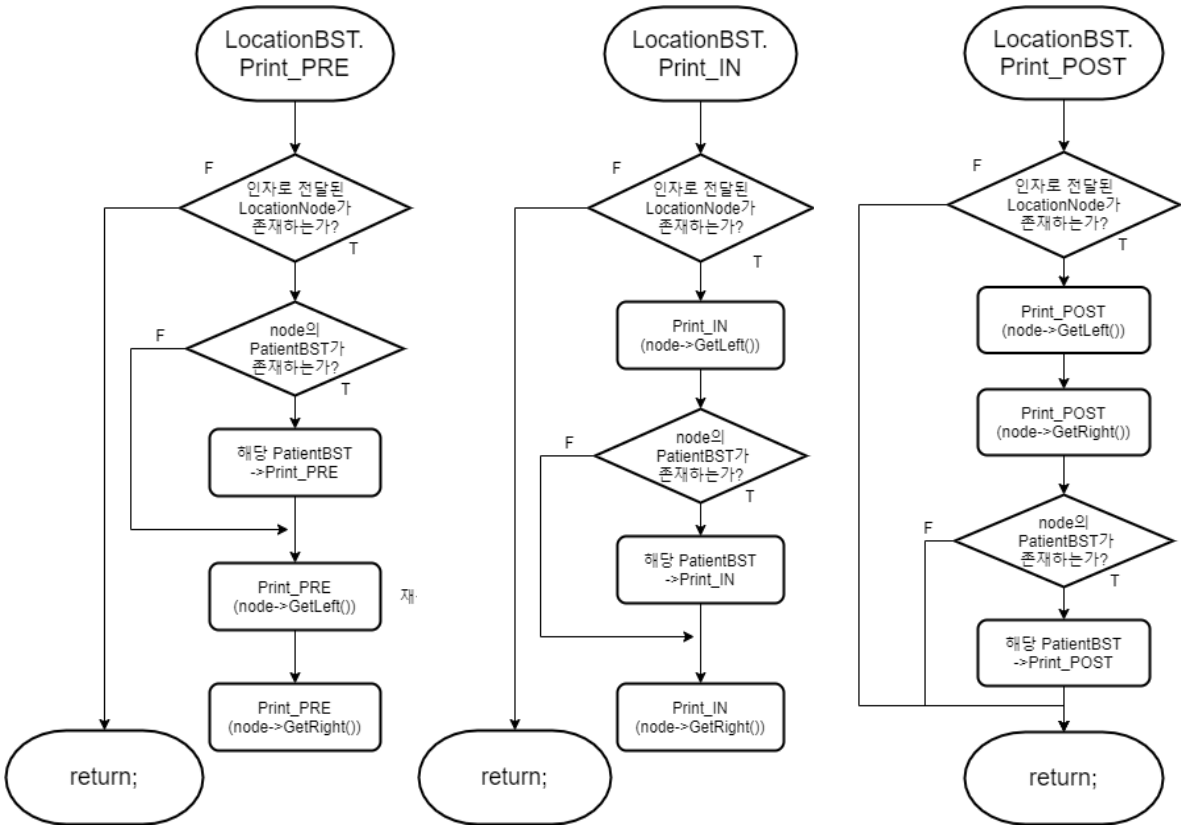
A. Manager.PRINT()

command.txt에서 읽은 PRINT명령어를 수행하는 함수이다. 이 PRINT명령어는 두 번째 인자가 'B'인 경우 출력방법을 세 번째 인자로 받고, 두 번째 인자가 'H'인 경우 세 번째 인자는 받지 않는다. Manager.PRINT함수는 명령어의 인자를 확인하여 해당 함수를 호출한다.



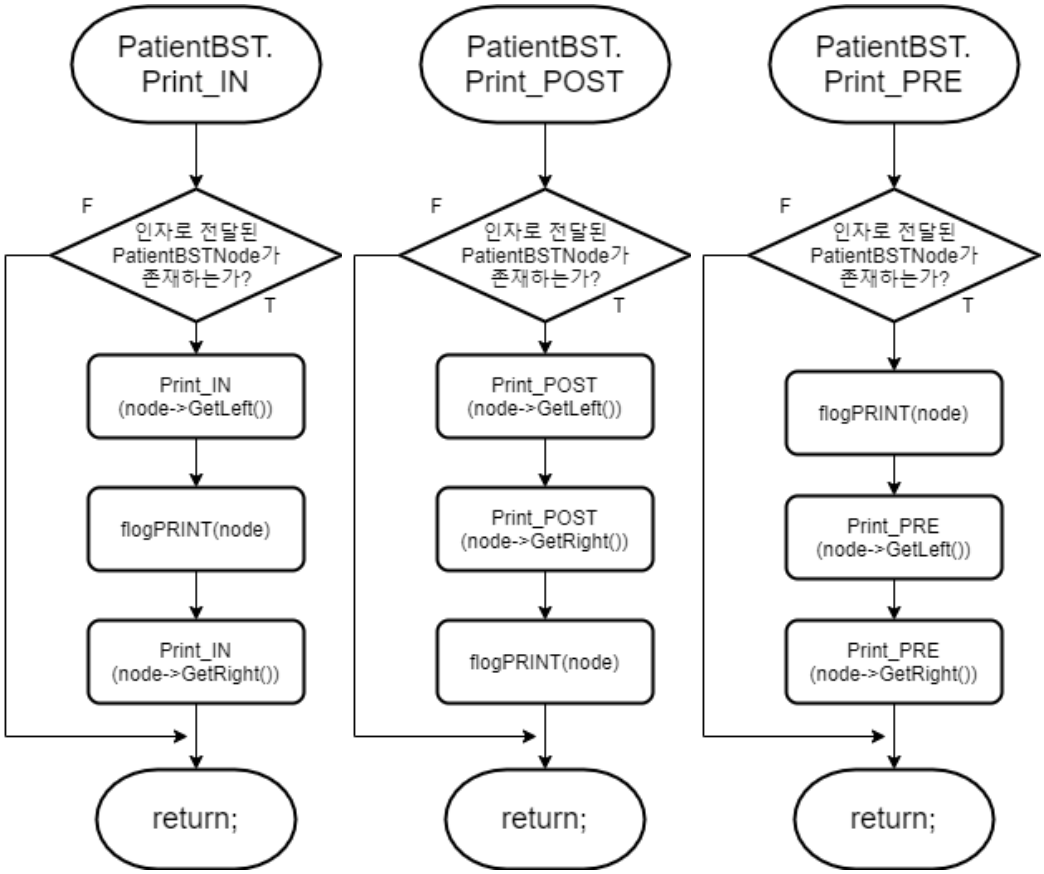
B. LocationBST.Print_PRE(), IN(), POST()

LocationBST의 PRINT_PRE, PRINT_IN, PRINT_POST함수는 각각 pre order, in order, post order 방법으로 LocationBST를 순회한다. LocationNode를 인자로 받아 재귀호출되어 사용된다.



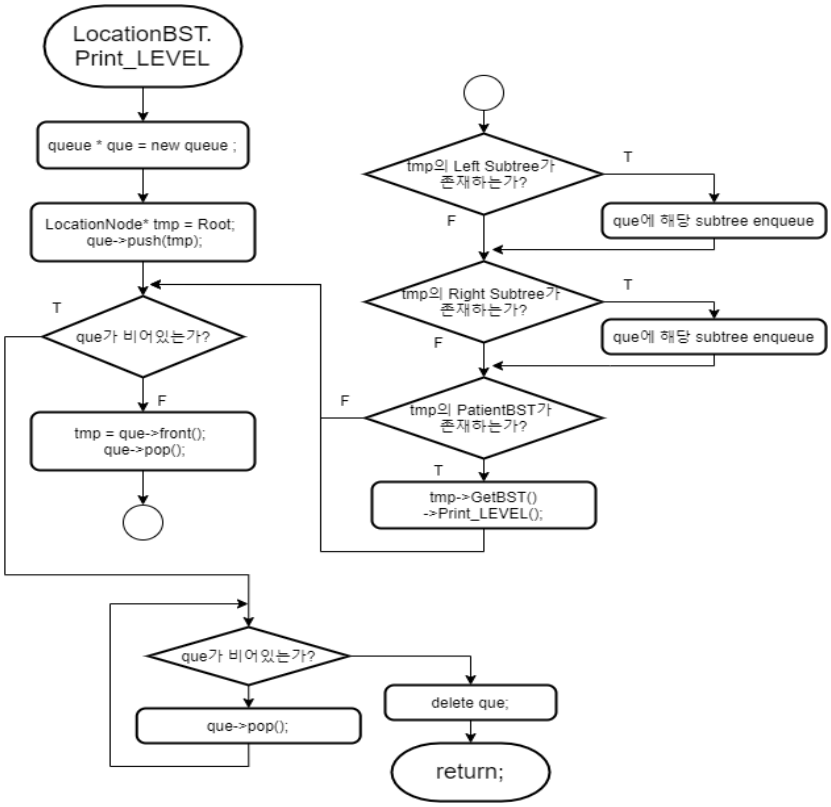
C. PatientBST.Print_PRE(), IN(), POST()

LocationBST의 Print_PRE, Print_IN, Print_POST 함수에 의해 각각의 함수가 호출된다. 순회 방법에 맞는 순서로 flogPRINT함수를 호출하여 PatientBST의 내용을 log.txt파일에 출력한다.



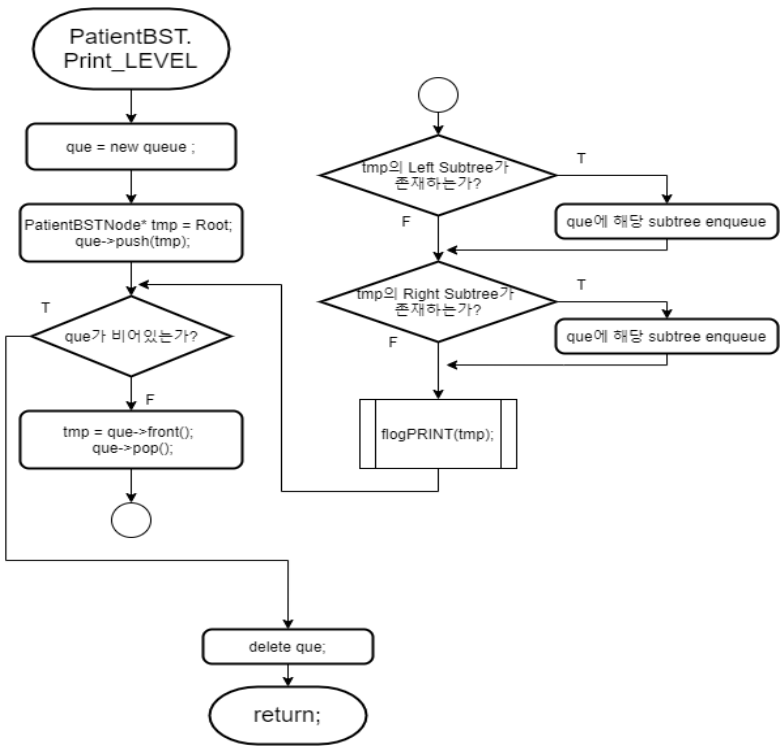
D. LocationBST.Print_LEVEL()

LocationBST를 LEVEL 순회하는 함수이다. Queue에 root node를 넣어서 시작하며, queue의 요소를 dequeue할 때, 해당 node의 Left, Right child를 queue에 enqueue한다.



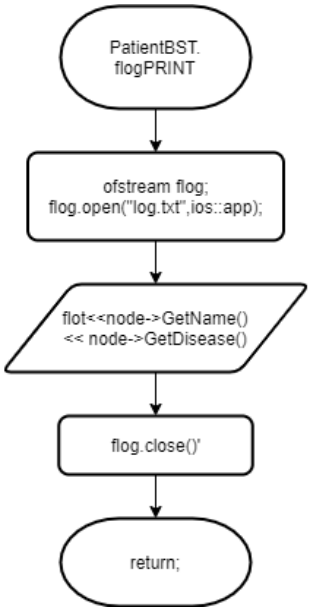
E. PatientBST.Print_LEVEL()

LocationBST의 Print_LEVEL함수와 유사한 구조를 지닌다. 동일한 방법으로 PatientBST를 순회하고, flogPRINT함수를 호출하여 값을 출력한다.



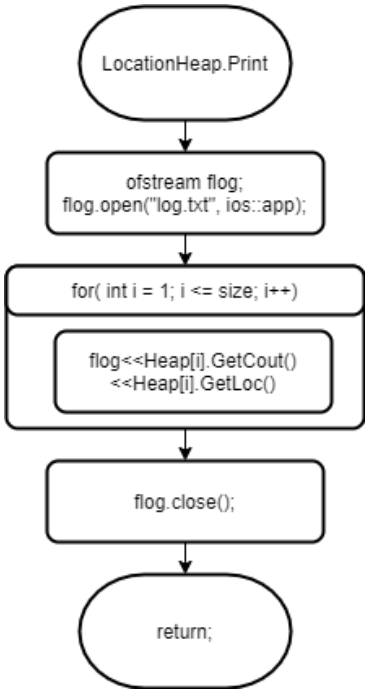
F. PatientBST.flogPRINT()

PatientBSTNode를 인자로 받아 log.txt파일에 노드의 정보를 출력한다.



G. LocationHeap.Print()

LocationHeap에 존재하는 모든 정보를 출력한다. MaxHeap구조를 배열로 구현했으므로 for반복문을 이용하여 출력할 수 있다.

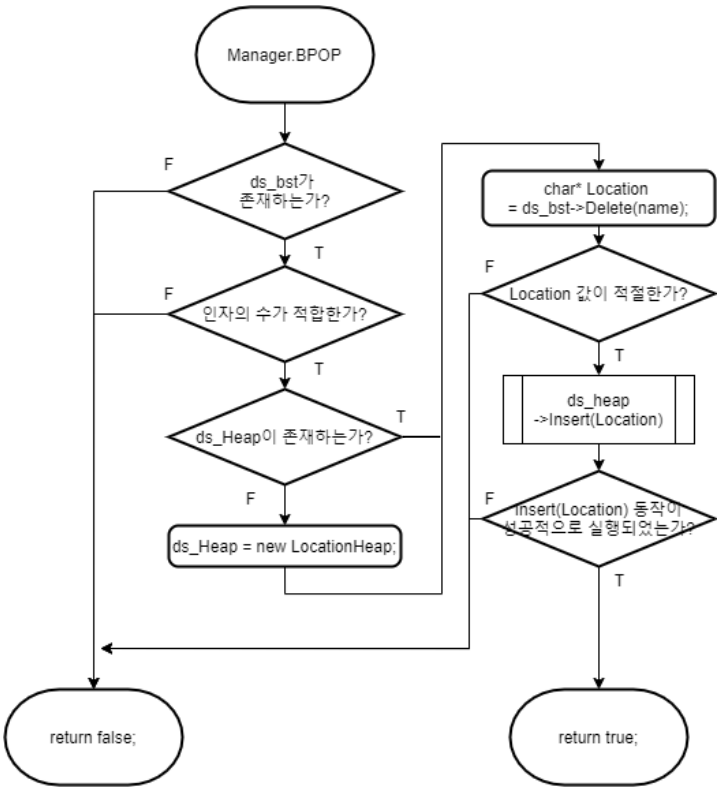


---

(7) BPOP

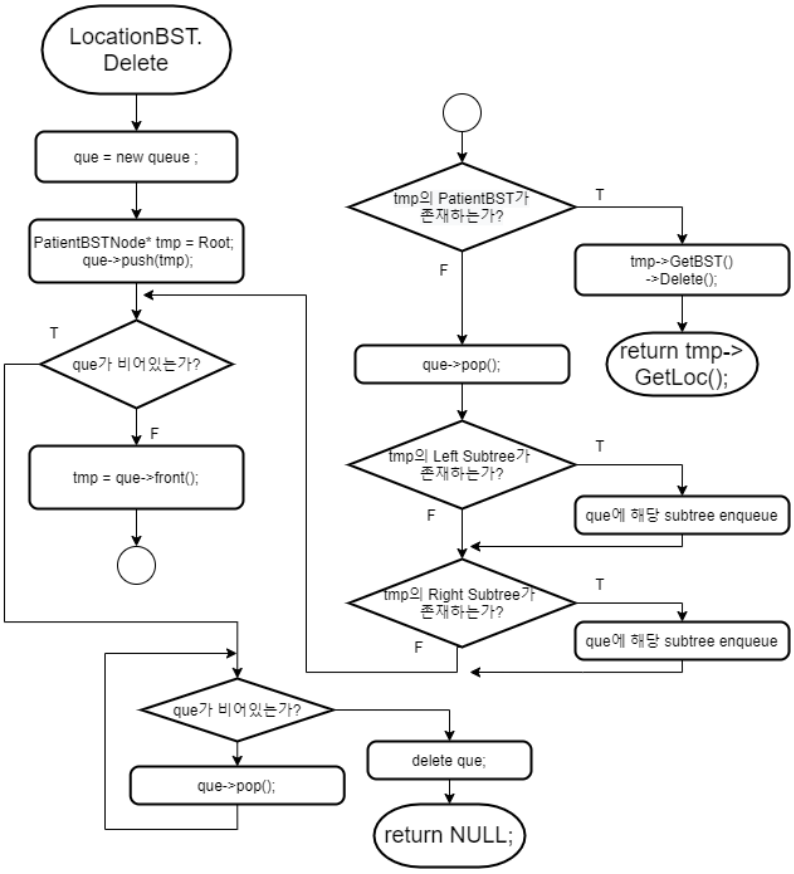
A. Manager.BPOP()

LocationBST에서 환자의 이름을 통해 해당 노드를 찾아내어 BST에서 삭제하고, maxheap에 삽입한다. 인자의 수가 적절한지 여부와 BST의 존재여부를 확인하고, maxheap이 없는 경우 새로 생성한다. LocationBST의 Delete함수를 이용하여 bst에서 특정 환자의 노드를 삭제하고 LocationHeap의 Insert 함수를 이용해 새 노드를 삽입한다.



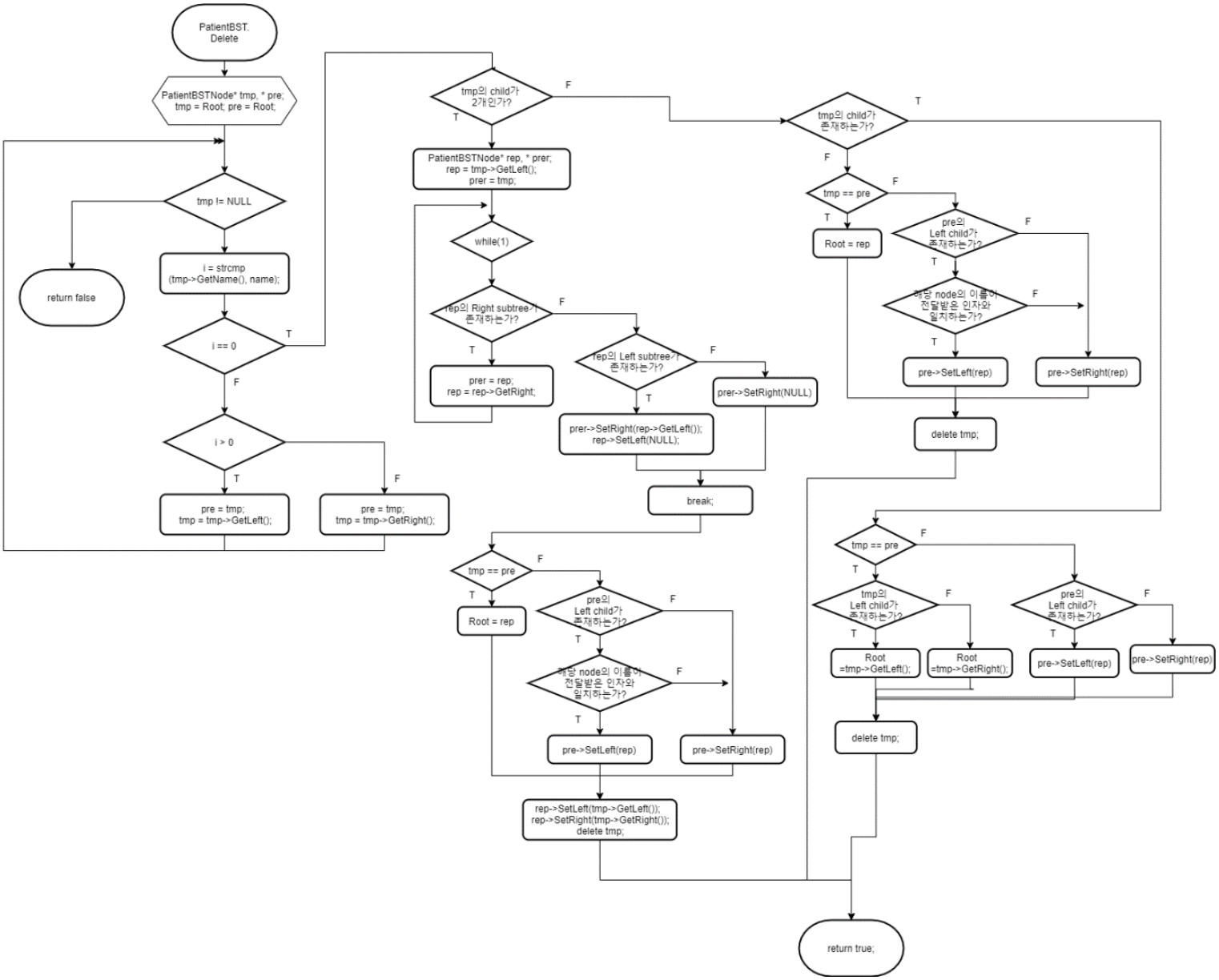
B. LocationBST.Delete()

Level traversal을 이용하여 인자로 전달받은 이름과 동일한 이름을 가진 노드를 검색하여 삭제한다. 삭제에 성공한 경우 해당 노드의 지역명을 반환한다.



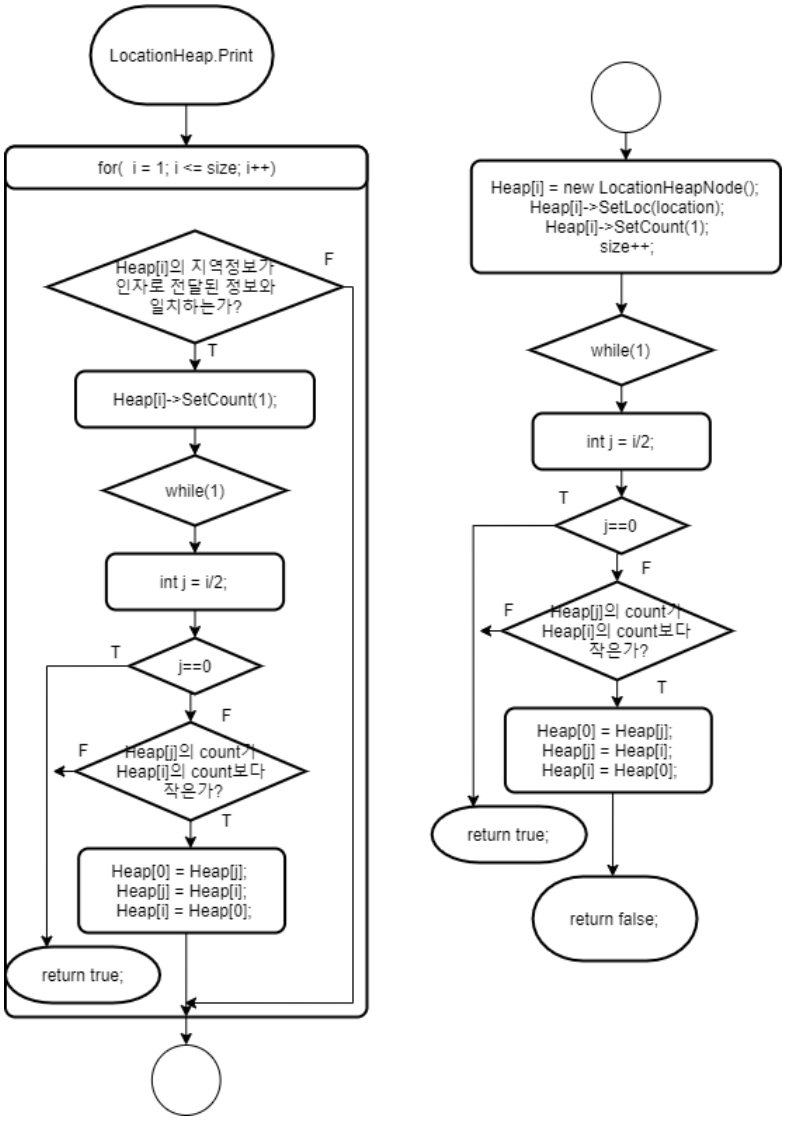
C. PatientBST.Delete()

LocationBST의 Delete 함수에 의해 호출되어 노드를 삭제하는 동작을 수행한다. 삭제할 노드와 그 부모노드의 하위 node 수에 따라 경우를 나누어 실행된다.



D. LocationHeap.Insert()

PatientBSTNode를 삭제하고 얻은 정보를 바탕으로 LocationHeapNode를 생성한 뒤, LocationHeap에 삽입하고 재정렬하는 함수이다. Maxheap의 특징을 이용하여 while반복문을 이용해 정렬한다. 인자로 전달된 지역의 LocationHeapNode가 없는 경우 새로 생성하고 Maxheap을 정렬한다.



---

3. Algorithm: 프로젝트에서 사용한 알고리즘

(1) 순회알고리즘 (PRE/LEVEL)

BST를 순회하는 알고리즘 중 PRE, IN POST order 순회의 구조는 유사하므로 PRE order 순회의 예로 모두를 설명하기로 한다.

PRE order traversal을 하는 함수는 재귀 호출되어 사용된다. 인자로 순회하는 BST의 Node의 포인터를 받고, 해당 node가 존재하는지 확인한다. 그 후에 PRE order순회 순서에 맞도록 인자로 전달된 node의 정보를 조회하고, 해당

node의 Left child node를 인자로 전달하여 다시 PRE order 순회 함수를 호출한다. Left child node를 인자로 한 함수의 동작이 종료되면, 다음으로 Right child node를 인자로 하여 다시 같은 함수를 호출한다. 이 과정을 통해 BST의 모든 node를 PRE order 순회할 수 있다.

IN order 순회의 경우 인자로 전달된 node를 조회하기 전에 Left child node를 인자로 한 In order 순회 함수를 호출하는 것이 먼저 실행되고, POST order 순회는 자기자신 node의 정보를 조회하는 것이 가장 마지막 순서가 된다.

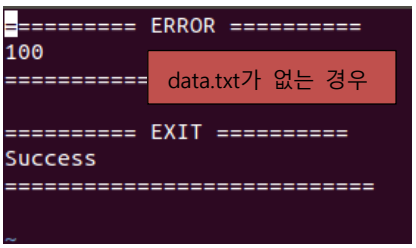
(2) MaxHEAP 정렬 알고리즘

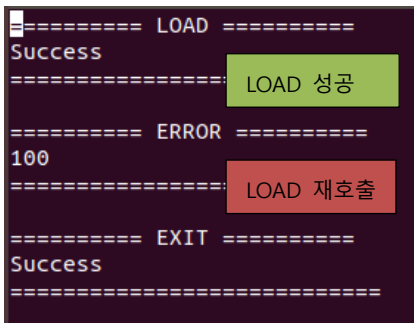
Maxheap은 자식 노드의 값이 부모 노드의 값보다 항상 작거나 같은 complete Binary tree이다. 이 경우 모든 노드가 순서대로 채워지므로 배열을 이용하여 구현할 수 있다. BPOP명령어를 통해 LocationHeapNode가 추가되거나 값의 변경이 생긴 경우, Heap을 재정렬 해야한다.

값이 변경된 노드가 i번째 노드일 때, 해당 노드의 부모 노드는 i/2번째 노드이다. 변경된 값이 부모 노드의 값보다 큰 경우에만 순서변경이 발생하므로, 해당 경우를 확인한다. Heap[i]의 값이 Heap[i/2] 보다 큰 경우, 두 노드의 순서를 뒤집고, i에 i/2값을 저장한 뒤 다시 부모 노드와 비교한다.

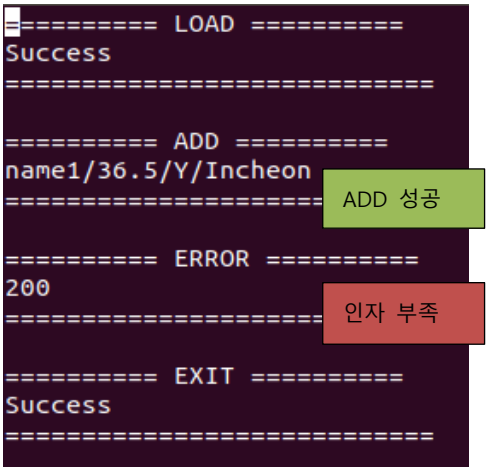
4. Result Screen: 모든 명령어에 대한 결과화면

(1) LOAD

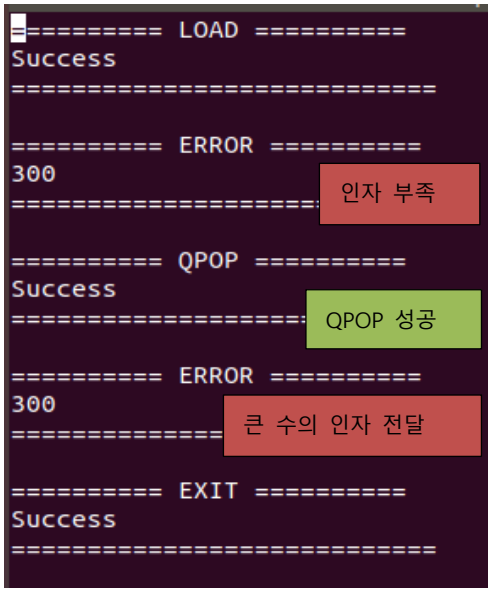




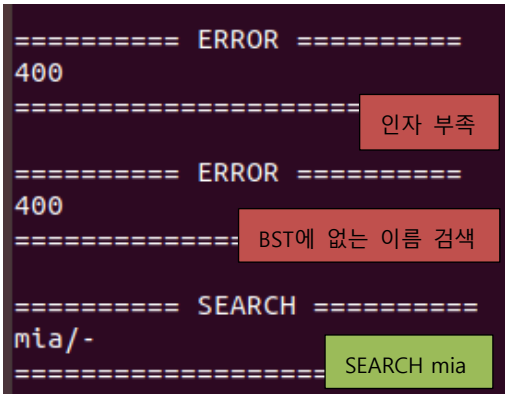
(2) ADD



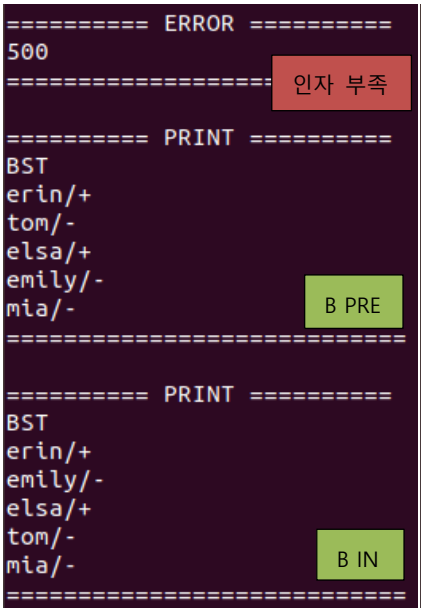
(3) QPOP

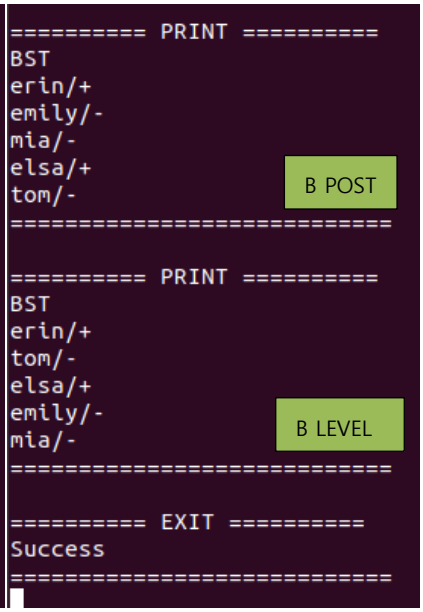


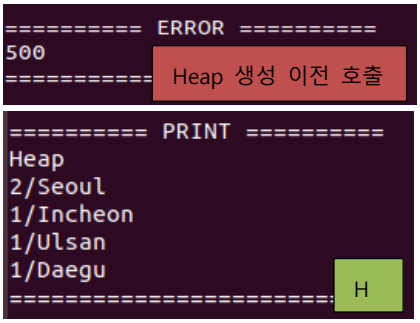
(4)SEARCH



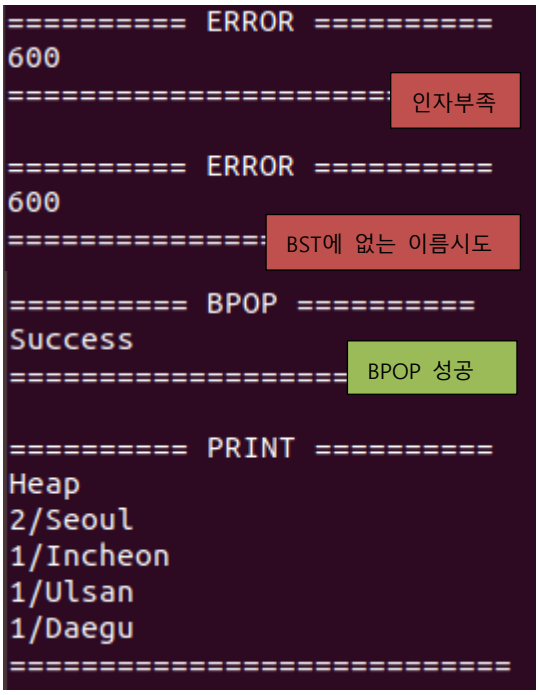
(5)PRINT







(6) BPOP



5. Consideration: 고찰

수업시간에 배운 자료구조를 직접 구현하여 문제를 해결하는 과제였다. 이번 과제에서는 queue, BST, Maxheap 자료구조를 이용하였다. LocationBST의 node가 또 다른 BST인 PatientBST를 포함하는 구조가 다소 복잡하게 느껴졌다.

자료구조를 학습할 때와는 다르게 node간의 연결관계를 코드로 표현하고, 디버깅 과정에서 확인하는 것이 시각적으로 표현한 것과 크게 달라 확인이 어려웠다. 이 경우 node의 자식 node들을 하나씩 확인하며 node간의 연결관계를 그림으로 그려서 전체 BST의 구조를 파악하였다.

프로그램을 작성하는데 사용된 일부 함수의 경우 함수의 길이가 과도하게 길어져서 수정과 분석에 어려움을 겪었다. 중복되는 부분을 새로운 함수로 정의하여 사용하면 이런 문제점을 해결할 수 있을 것이다. 다음부터 문제를 풀 때 프로그램의 수정까지 고려하여 코드를 작성하는 것이 좋을 것 같다.