제출일 : 2017. 9. 29

데이터구조설계

〈PROJECT#1〉

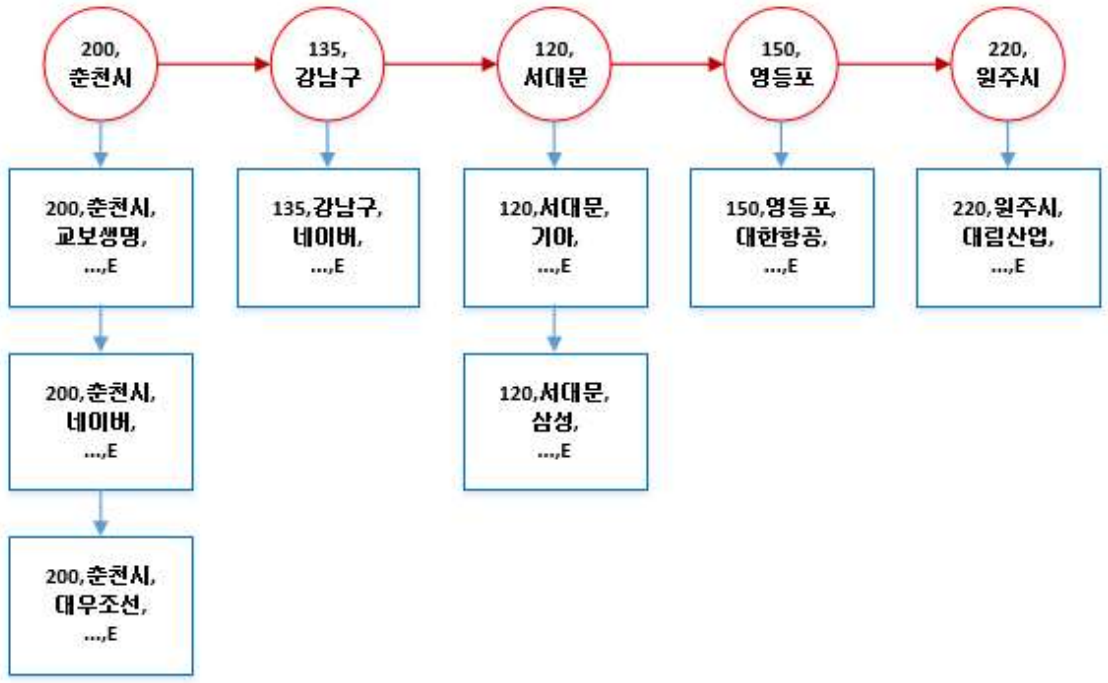
담당교수	**이기훈 교수님**
학 과	**컴퓨터공학과**
학 번	**2014722075**
이 름	**이 동 준**

KWANGWOON UNIVERSITY

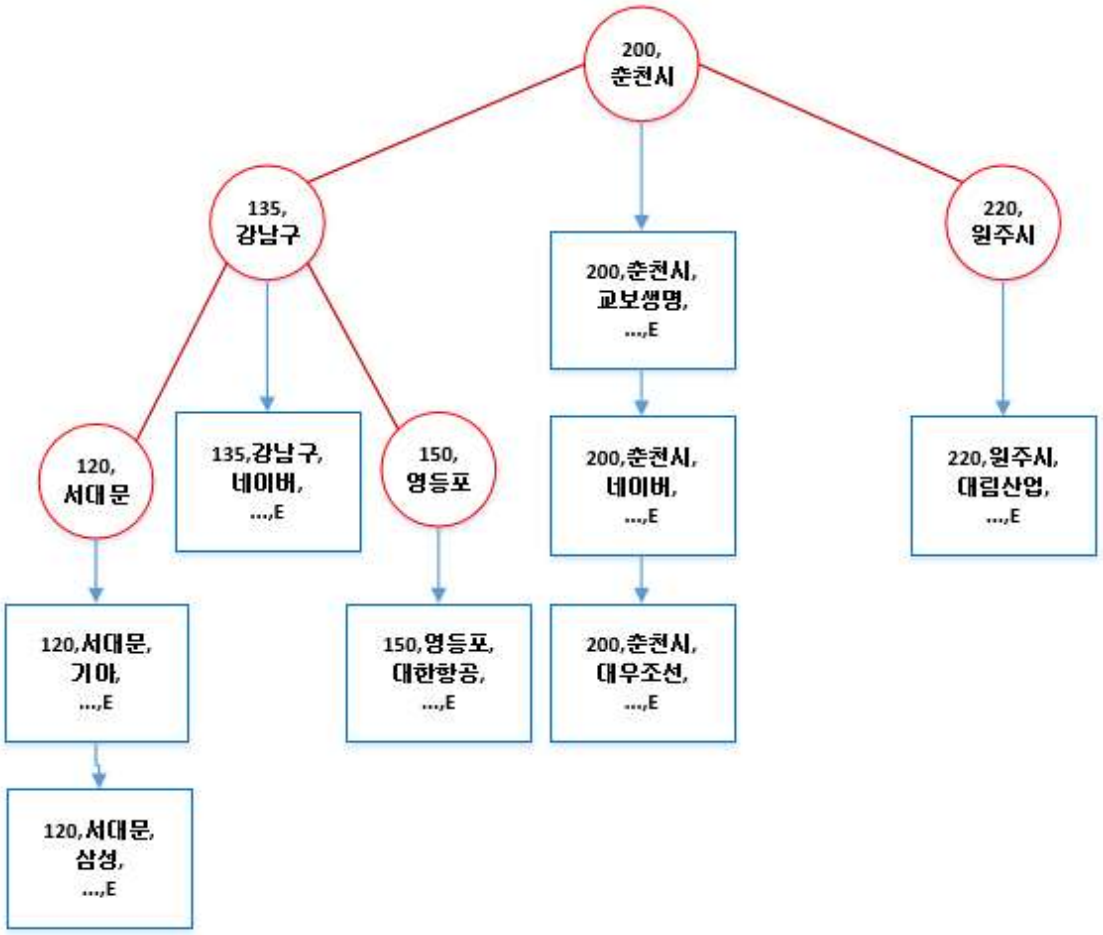
● Introduction

- 이 프로젝트는 이진트리탐색(BST)와 2D Linked List, Circular Linked List를 이용하여 지역별 채용정보 관리 프로그램이다. 채용정보에는 지역코드, 지역이름, 기업명, 지원자격, 근무조건, TO가 있고 여기서의 TO란 채용중인 상태이면 'E(exist)', 채용이 마감상태 이면 'NE(not exist)'를 뜻한다. TO의 상태가 'E'면 2D Linked list에 입력되고 모든 정보들이 입력된 후에는 2D Linked List를 기반으로 BST를 구성한다. 반면 'NE'의 상태라면 Circular Linked List에 입력되어 관리된다. 2D Linked List와 BST에서는 Region Node와 Employment Node, 두 가지의 노드를 사용하게 되는데 Region Node는 지역코드와 지역이름을 가지고 있는 노드로, 2D Linked List에서는 횡의 List를 뜻하고 BST에서는 Treenode를 뜻하게 된다. 이에 따라 입력되는 지역코드와 지역이름에 따라 모든 채용 data를 가지고 있는 Employment Node는 Region Node 아래, 즉 종의 List를 뜻한다. 하지만 EmployNode중 TO의 상태가 'NE'인 경우에는 Circular Linked List에서 Region Node 없이 삽입하게 된다. 명령어에는 LOAD, UPDATE, FIND, DELETE, PRINT, EXIT가 있고 command파일에서 명령어를 실행 시킨다.

- 2D linked list 예시



- BST 예시



- Circular linked list 예시

- 명령어 설명 : Command text파일에 입력되어 있는 명령어들을 차례로 실행

1. LOAD : “IN_LOAD.txt“ text파일에 저장 되어있는 채용 정보들을 읽어와 상태에 따라 2D Linked List, BST 그리고 Circular Linked List에 저장한다.

-error-

100-1 : 텍스트 파일이 존재하지 않는 경우

100-2 : 같은 지역에 기업명이 중복된 데이터를 삽입하는 경우

2. UPDATE : "IN_UPDATE.txt" text파일에는 LOAD에서 읽어온 DATA의 수정판이 담겨져 있는데 TO를 'E'에서 'NE'로 혹은 'NE'에서 'E'로 변형이 된다. 이에 따라서 Data를 관리하고 있는 기존의 List에서 알맞은 List로 이동을 한다. 만약 2D linked list에서 마지막 employment node가 E->NE대상이라면 Region node는 필요없어지기 때문에 그 Region node는 삭제를 한다. 마찬가지로 NE->E 대상 중 Region node가 존재하지 않다면 2D linked list에 insert전에 해당 employment node의 지역코드과 지역명을 가진 Region node를 생성해준다.

   -error-

   200-1 : 텍스트 파일이 존재하지 않는 경우

   200-2 : 기존 데이터에 존재하지 않는 채용정보를 Update하는 경우

3. FIND : 입력된 지역코드의 채용 중인 정보를 찾는 명령어. BST의 지역코드를 탐색하여 모든 채용 중인 정보를 출력한다.

   -error-

   300-1 : 해당지역에 채용 중인 정보가 존재하지 않으면 에러 출력

4. DELETE : 입력된 지역코드의 채용 중, 채용마감 정보를 모두 삭제하는 명령어. 지역 코드를 가지고 있는 모든 2D Linked List, BST 그리고 Circular Linked List의 Region Node와 Employmend Node를 메모리 해제한다.

   -error-

   400-1 : 해당지역에 채용 중 및 채용마감인 정보가 존재하지 않는 경우

5. PRINT : 현재 채용중인 정보를 모두 출력하는 명령어. BST에서 In-order traversal방법 즉, 지역코드가 작은 것부터 출력을 하게 된다. 위 In-order traversal방식에서 stack을 사용하게 되는데, stack이란 쓰레기통과 같다 먼저 들어온 것이 맨 밑에부터 누적이되고, 출력은 가장 마지막에 들어온 것부터 출력된다. root노드부터 진행하여 왼쪽 서브트리를 모두 출력하고 오른쪽 서브트리를 출력한다.

   -error-

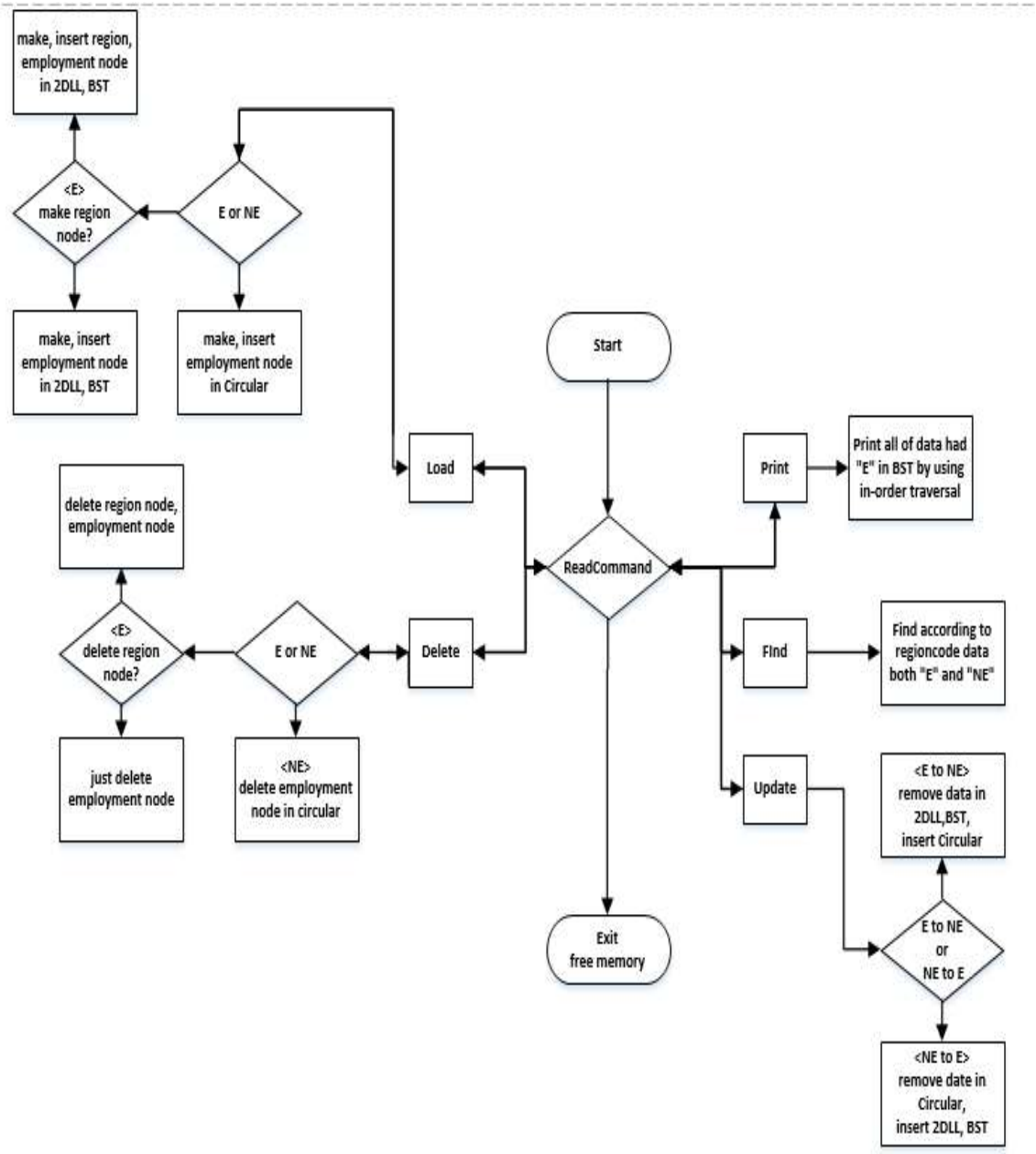
   500-1 : 채용 중인 정보가 존재하지 않을 경우

- Node 설명

1. Region Node : 지역 코드와 지역 이름을 가지고 있는 노드로, 2D linked list의 횡에 쓰이고 BST에서는 각 레벨의 node로 사용 된다.

2. Employment Node : 지역코드, 지역명, 기업명, 지원자격, 근무조건, TO의 정보를 가지고 있고, TO가 E인 노드는 같은 지역코드를 가지 region node 아래에 기업명 순으로 위치하게 되고, TO가 NE인 노드는 Circular linked list로 관리된다.

● Flowchart



readcommand를 중심으로 각 명령어에 따라 실행을 한다. 실행 후에는 다음 command를 입력받아 명령어를 실행한다. EXIT명령어가 실행되면 프로그램을 종료하는 동시에 모든 메모리를 해제한다.

● Algorithm

```
# ReadCommand
        file.open(command text file)
        if load
                file.open(load text file)
                m_2DLL,m_BST = data TO is “E”
                m_Circular = data TO is “NE”
        if find
                show all of data had some region code
                file.cout(output file)
        if delete
                find data
                if nodes should deleted were in 2DLL, BST
                        delete region,employment node in 2DLL, BST
                if nodes should deleted were in Circular
                        delete employment node in Circular
                file.cout(output file)
        if update
                file.open(update file)
                find employment nodes will be updated
                if E to NE
                        update E -> NE
                        if region node must be deleted
                                remove region, employment node in 2DLL, BST
                                send employment node to Circular
                        else
                                send employment node to Circular
                else (NE to E)
                        update NE -> E
                        if region node must be created
                                make region, employment node
                                insert to 2DLL, BST
                        else
                                insert to 2DLL, BST
        if print
                show all of data in BST by using in-order traversal
                file.cout(output file)
        if exit
                free memoty
                exit program
```

2D linked list

- Insert(Region node) - insert rNode
    if pHead = NULL
        sethead(rNode)
    else
        tailnode->setnext(rNode)
- Insert(Employment node) - insert eNode
    find Region node same region code
    setdown eNode(must be sorted according to company
- Delete - use regioncode
    find region node had region code
    delete region node and all of employment node
    revise linked list

BST

- Insert - using 2D linked list
    root = 2DLL->gethead
    insert region node according to region code in BST shape
- Search - using region code
    search region node by comparing region code
    return employment node under the region node
- Delete - use regioncode - delete p
    find region node
    if p is leaf
        delete p
    if p only has right child
        displace p to right child
    is p only has left child
        displace p to left child
    if p has left child and right child
        displace p to biggest node in left subtree of p

Circular linked list

- Insert - insert eNode
    if pHead = NULL
        sethead(eNode)
    else
        tailnode->setnext(eNode)
        eNode->setnext(pHead)
- Delete - delete p
    find p
    delete p and revise linked list

● Result Screen

- LOAD

```
======= LOAD ========
IN_LOAD.txt LOAD Successtul
=============================
```

-> LOAD 파일을 열어서 (이때 로드파일이 없을시 에러 출력), 그 안에 있는 채용 정보들을 하나씩 받아와 E와 NE 즉, TO로 구분하여 E는 2D linked list에 NE는 Circular에 insert해준다(만약 같은 지역에 같은 회사명이 존재하면 에러 출력). 이때 2D linked list의 region node와 Circular의 모든 노드들은 insert 순서대로 입력된다. 하지만 2D linked list의 employment node는 기업명에 따라 sort되어야 한다. 2D linked list의 region node를 토대로 BST를 설계한다.

- FIND

```
=====FIND - 158 지역의 채용중인 기업======
지역코드,지역명,기업명,지원자격,근무조건,TO
=========================================================
158,양천구,엘지화학,석사이상,계약직,E
158,양천구,지에스칼텍스,고졸이상,인턴직,E
158,양천구,현대모비스,석사이상,정규직,E
158,양천구,현대중공업,석사이상,정규직,E

=====FIND - 200 지역의 채용중인 기업======
지역코드,지역명,기업명,지원자격,근무조건,TO
=========================================================
200,춘천시,교보생명보험,고졸이상,정규직,E
200,춘천시,네이버,석사이상,정규직,E
200,춘천시,대우조선해양,대졸이상,정규직,E
200,춘천시,하나은행,대졸이상,인턴직,E
```

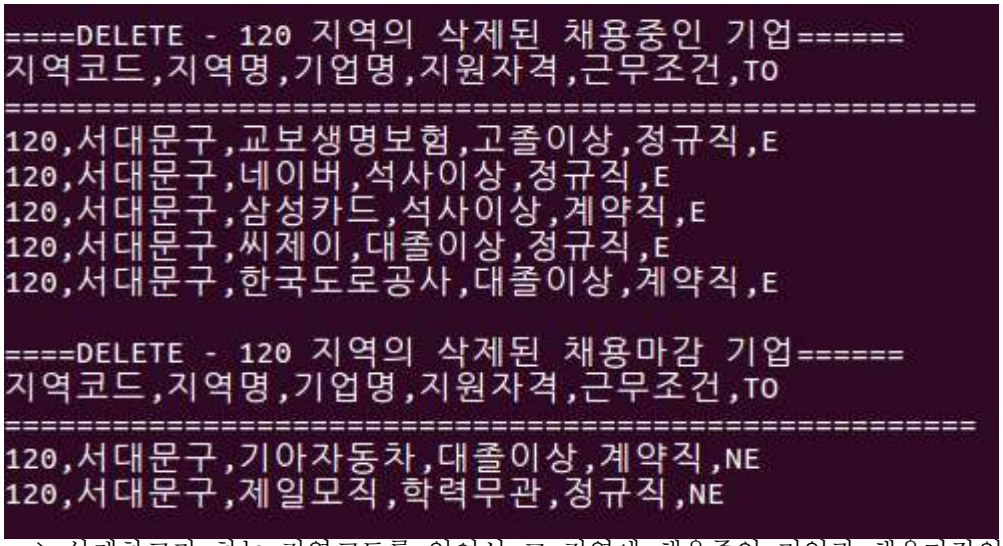
-> 찾고자 하는 지역의 코드를 받아와 그 지역의 현재 채용중인 기업을 출력한다(이때 찾으려는 지역의 채용중인 정보가 존재하지 않으면 에러 출력).

- UPDATE

```
======== ERROR ========
200-2
======================
============UPDATE - 변경된 채용정보=============
지역코드, 지역명, 기업명, 지원자격, 근무조건, TO
=========================================================
158,양천구,지에스칼텍스,고졸이상,인턴직,NE
158,양천구,현대모비스,석사이상,정규직,NE
158,양천구,하이닉스,석사이상,정규직,E
158,양천구,제일모직,학력무관,정규직,E
200,춘천시,교보생명보험,고졸이상,정규직,NE
200,춘천시,네이버,석사이상,정규직,NE
200,춘천시,하이닉스,석사이상,정규직,E
```

-> update파일을 열어 (이때 update파일이 존재하지 않는 경우 에러 출력), update하려는 data들을 받아오고 E to NE인지 NE to E인지를 구별하여 2DLL, BST에서 Circular인지 Circular에서 2DLL, BST인지를 결정한다(만약 update하려는 정보가 존재하지 않으면 에러 출력). 더불어 2DLL, BST에서 employment node가 하나도 남지 않게 되는 경우에는 region node는 삭제된다. 반대로 Circular에서 2DLL로 이동할 때 region node가 존재하지 않으면 새로운 region node를 삽입한다.

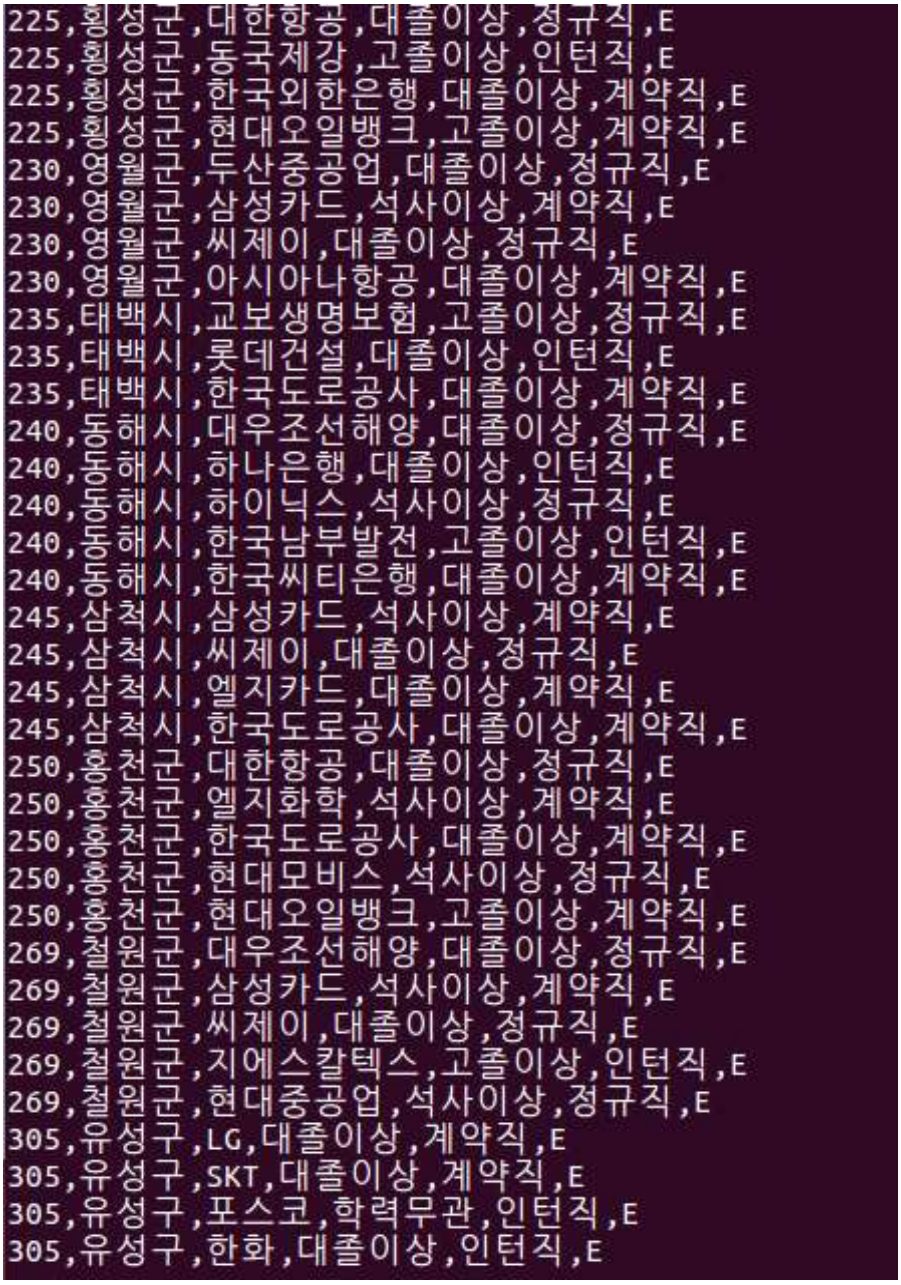
- DELETE



-> 삭제하고자 하는 지역코드를 읽어서 그 지역에 채용중인 기업과 채용마감인 기업을 모두 삭제한다(이때 삭제하려는 정보가 양쪽 모두 없으면 에러 출력). BST같은 경우에는 삭제하려는 노드의 자식이 두 개 있을 경우 왼쪽 subtree에서 가장 큰 region노드로 대체를 한다.

- PRINT

```
=====PRINT -  채용중인 기업========
지역코드,지역명,기업명,지원자격,근무조건,TO
=========================================================
110,종로구,LG,대졸이상,계약직,E
110,종로구,SKT,대졸이상,계약직,E
110,종로구,포스코,학력무관,인턴직,E
130,동대문구,우리은행,경력무관,인턴직,E
130,동대문구,한국가스공사,대졸이상,계약직,E
130,동대문구,현대중공업,석사이상,정규직,E
135,강남구,교보생명보험,고졸이상,정규직,E
135,강남구,네이버,석사이상,정규직,E
135,강남구,대우조선해양,대졸이상,정규직,E
135,강남구,씨제이,대졸이상,정규직,E
135,강남구,지에스칼텍스,고졸이상,인턴직,E
135,강남구,한국도로공사,대졸이상,계약직,E
140,용산구,신한은행,대졸이상,정규직,E
140,용산구,한국외한은행,대졸이상,계약직,E
150,영등포구,대한항공,대졸이상,정규직,E
150,영등포구,현대오일뱅크,고졸이상,계약직,E
158,양천구,엘지화학,석사이상,계약직,E
158,양천구,제일모직,학력무관,정규직,E
158,양천구,하이닉스,석사이상,정규직,E
158,양천구,현대중공업,석사이상,정규직,E
200,춘천시,대우조선해양,대졸이상,정규직,E
200,춘천시,하나은행,대졸이상,인턴직,E
200,춘천시,하이닉스,석사이상,정규직,E
210,강릉시,대우건설,고졸이상,인턴직,E
210,강릉시,동국제강,고졸이상,인턴직,E
210,강릉시,두산중공업,대졸이상,정규직,E
210,강릉시,롯데건설,대졸이상,인턴직,E
210,강릉시,삼성중공업,대졸이상,정규직,E
210,강릉시,아시아나항공,대졸이상,계약직,E
210,강릉시,지에스건설,학력무관,계약직,E
215,양양군,교보생명보험,고졸이상,정규직,E
215,양양군,네이버,석사이상,정규직,E
215,양양군,대우조선해양,대졸이상,정규직,E
215,양양군,삼성카드,석사이상,계약직,E
215,양양군,씨제이,대졸이상,정규직,E
215,양양군,한국도로공사,대졸이상,계약직,E
220,원주시,대림산업,대졸이상,계약직,E
220,원주시,대한항공,대졸이상,정규직,E
220,원주시,삼성테스코,석사이상,정규직,E
220,원주시,쌍용자동차,석사이상,계약직,E
220,원주시,엘지화학,석사이상,계약직,E
220,원주시,현대모비스,석사이상,정규직,E
220,원주시,현대오일뱅크,고졸이상,계약직,E
```



-> BST에서 stack과 in-order traversal기법을 이용하여 입력되어 있는 채용중인 기업의 정보를 모두 출력한다(이때 채용중인 정보가 존재하지 않을 경우 에러 출력).

● Consideration

이번 프로젝트를 진행하면서 평소 어려웠던 클래스 활용에 대해 다시 한번 연습할 수 있어서 좋았다. 클래스를 선언해주는 것부터 클래스의 활용법까지 새로히 알 수 있었다. 입대전 2학년 1학기와는 난이도가 많이 차이 났었고, 프로젝트의 스케일에서 처음에는 내가 할 수 있을까? 라는 의문점을 가졌지만, 방학 중 연습했던 것에서 나와 어쩌면 행운이였다. 데이터구조설계 강의에서 처음으로 BST를 접하게 되었는데 개념자체는 이해하기 쉬웠는데 코드로 작성을 하려니 막막하였다. 강의자료를 참고하면서 하나하나씩 코드를 써내려갔고 수정의 과정도 많았다. 특히 DELETE와 UPDATE에서 자식노드가 두 개인 경우에는 대체

해야 할 노드를 찾아 대체하는 것이 가장 어려운 부분이었고, UPDATE에서도 마찬가지였다. 여러 가지 case가 있었고 생각하면 할수록 다양한 케이스가 등장해 당황하였다. 코드를 작성해가면서 어떻게 하면 완성도가 높은 프로그램을 만들 수 있을지 고민을 많이 하였고 다양한 시도도 해보았다. 이번 프로젝트를 하면서 다시 한번 클래스를 익혀 매우 유익했고, 2차 프로젝트도 노력하여 완성을 시키고 싶다.