

Due : ~2023.04.12.(Wed) 23:59, Late Submission : ~2023.04.13.(Thu) 23:59

- **Binary Search Tree Implementation - Insert, Delete, Find, PrintInorder.**
- 아래의 사진과 같이 **[input].txt** 파일을 입력받아 **BST**를 생성하고 편집하는 기능을 구현하고, 각 커맨드에 알맞은 결과 값을 **[output].txt** 파일에 출력하여 저장한다.
- **Implement the function of creating and editing a BST by receiving the [input].txt file as input as below and the output result of input commands is stored in the [output].txt file.**

```

input.txt
i 1
i 2
i 3
i 5
i 5
pi
f 5
f 4
d 5
d 5
pi

output.txt
insert 1
insert 2
insert 3
insert 5
insertion error: 5 is already in the tree
1 2 3 5
5 is in the tree
finding error: 4 is not in the tree
delete 5
deletion error: 5 is not in the tree
1 2 3
  
```

<input> :

- 각 line 마다 Command 가 주어짐.(insert, delete, print, find)
- **insert** : 위의 사진에서처럼 **"i [insert할 key값]"** 로 표현합니다.
- **delete** : 위의 사진에서처럼 **"d [delete할 key값]"** 로 표현합니다.
- **print** : 위 사진처럼 **"pi"** 로 표현 합니다.
- **find** : 위의 사진처럼 **"f [search할 key값]"** 로 표현합니다.
- Commands are given line by line (insert, delete, print, find)
- **insert** : Expressed as **"i [key value to be inserted]"** as in the picture above
- **delete** : Expressed as **"d [key value to be deleted]"** as in the picture above
- **print** : Expressed as **"pi"** as in the picture above
- **find** : Expressed as **"f [key value to be search]"** as in the picture above

<output> :

- 각 **Command** 에 맞는 **result** 출력.
- **insert**가 수행되면 **"insert key값\n"**, 이미 **key**가 존재했으면, **"insertion error: key값 is already in the tree\n"** 를 출력합니다.
- **delete**가 수행되면 **"delete key값\n"**, delete할 **key**를 찾을 수 없으면, **"deletion error: key값 is not in the tree\n"**를 출력합니다.
- **tree**가 비어 **print** 할 값이 존재 하지 않으면, **"tree is empty"** 출력, 그렇지 않으면, **"key1 key2 key3 ..."** 을 출력, **key값** 들 사이의 한칸의 공백을 둘 것. (마지막 숫자 이후 공백없이 줄바꿈.)
- **입력으로 받은 f**로 **find**를 수행했을 때, 찾았으면 **"key is in the tree\n"**를 출력, 찾지 못했으면, **"finding error: key값 is not in the tree\n"**출력합니다. 입력으로 받은 **f**를 구분한 이유는, **insert, delete**함수 실행을 위해 **main**함수에서 사용하는 **find**함수의 경우 해당 메시지를 출력하지 않기 위함입니다.
- **Appropriate output messages for each command.**

- If insert is executed, print "insert key value\n", and if the key already exists, print "insertion error: key value is already in the tree\n".
- If delete is executed, print "delete key value\n", and if the key to be deleted cannot be found, print "deletion error: key value is not in the tree\n".
- If the tree is empty and there is no value to print, print "tree is empty". otherwise, print "key1 key2 key3 ...", and leave a space between key values. (line break without spaces after the last number).
- When find is executed with f received as input, if the value is found, print "key is in the tree\n", and if the value is not found, print "finding error: key value is not in the tree\n". The reason for distinguishing the case f received as input with is not to print the message in the case of the find function used in the main function to execute the insert and delete functions

<insertNode>

- 입력한 Node들의 key값을 Binary Search Tree에 삽입하는 함수입니다.
- Insert 할 때마다, insert 할 노드를 출력시켜주셔야 합니다.
- "i 5" 의 경우 "5"노드를 BST에 추가하시고, "insert 5" 메시지를 출력시켜주시면 됩니다.
- Tree 안에 이미 존재하는 key값을 insert하는 경우 에러메시지를 출력해주셔야 합니다.
에러메시지 형식은 "insertion error: key값 is already in the tree" 입니다.
- Tree root : 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- int key : 삽입하려는 새로운 노드의 값입니다.
- Output : 새로운 노드가 삽입된 후 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- This function inserts the key values of the input nodes into the Binary Search Tree.
- "i [key value to be inserted]"
- For each insert, you must print out the node to be inserted.
- In the case of "i 5", node "5" should be inserted into BST, and the message "insert 5" should be printed out.
- When inserting the key value which already exists in BST, the error message should be printed out, in the format of "insertion error: key Value is already in the tree".
- Tree root : A pointer to the root node of the binary search tree.
- int key : The value of the new node to insert.
- Output : A pointer to the root node of the binary search tree after new nodes are inserted

<deleteNode>

- Delete 할 때마다, delete할 노드를 출력시켜주셔야 합니다.
- "d 5" 의 경우, Tree 안에 존재하는 "5"노드를 제거하시고, "delete 5" 메시지를 출력시켜 주시면 됩니다.
- Tree안에 존재하지 않는 key값을 제거하려 할 경우 에러 메시지를 출력해주셔야 합니다.
에러메시지 형식은 "deletion error: key값 is not in the tree" 입니다.
- Tree root : 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- int key : 삭제하려는 노드의 값입니다.
- Output : 삭제된 노드를 제거한 후 이진탐색 트리의 루트 노드를 나타내는 포인터입니다.
- Whenever you delete, print the node to be deleted

- "d **key value to be deleted**"
- For each delete, you must print out the node to be deleted.
- In the case of "d 5", node "5" should be deleted from BST, and the message "delete 5" should be printed out.
- When deleting the key value which does not exist in BST, the error message should be printed out, in the format of "**deletion error: key Value is not in the tree**".
- Tree root : A pointer to the root node of the binary search tree.
- int key : The value of the new node to be deleted.
- Output : A pointer to the root node of the binary search tree after deleting nodes.

<findNode>

- key값이 존재하는 경우, "**key값 is in the tree**" 출력합니다.
- key값이 존재하지 않는 경우, "**finding error: key값 is not in the tree**"로 출력해주셔야 합니다.
- Tree root : 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- int key : 찾으려고 하는 노드의 값입니다.
- Output : 찾으려고 하는 노드를 찾았으면 1, 찾지 못했으면, 0을 반환
- When the key value exists in the tree, "**key value is in the tree**",
- when the key value does not exist in the tree, "**finding error: key value is not in the tree**" should be printed out.
- Tree root : A pointer to the root node of the binary search tree.
- int key : The value of the node trying to find.
- Output : Returns 1 if the node is found, 0 if not found.

<printInorder>

- "pi"의 경우, Tree 전체를 Inorder로 순회하며 출력해주시면 됩니다.
- 출력 형식은 "**key값 key값 key값 ...**" 입니다. 모든 간격은 한칸 입니다. (마지막 숫자 이후 공백없이 줄바꿈.)
- Tree가 비어있을 경우, "**tree is empty**" 를 출력해주시면 됩니다.
- Tree root : 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- In the case of "pi", you can traverse the entire tree inorder and print it.
- In the case of "pi", the entire tree should be printed inorder.
- Printing format is "**key값 key값 key값 ...**", all the spaces are one space. (line break without spaces after the last number).
- If the tree is empty, "**tree is empty**" should be printed out.
- Tree root : A pointer to the root node of the binary search tree.

<deleteTree>

- 이진함수 트리를 완전히 삭제하는 기능을 수행합니다.
- Tree root: 이진 탐색 트리의 루트 노드를 나타내는 포인터입니다.
- Executes a function that completely deletes a binary function tree.
- Tree root : A pointer to the root node of the binary search tree.

<Structure & Function Format>

Structure

```
typedef struct BST* Tree;
typedef struct BST{
    int value;
    struct BST* left;
    struct BST* right;
}BST;
```

Function

```
Tree insertNode(Tree root, int key);
Tree deleteNode(Tree root, int key);
int findNode(Tree root, int key);
void printInorder(Tree root);
void deleteTree(Tree root);
```

- 위 사진과 같은 **Struct** 구조체를 사용하셔야 합니다.
- 위 사진과 같은 함수들을 형식에 맞게 구현해주시면 됩니다.
- **Struct format above should be used for implementation**
- **Functions should be implemented in appropriate format as above.**

<File Name Format>

- [StudentID].c ex) 20XXXXXXXXX.c

<Execution>

- gcc 20XXXXXXXXX.c -o 20XXXXXXXXX
- ./20XXXXXXXXX [input_file_name] [output_file_name]
- **!!! 꼭 제공되는 testcase로 실행시켜보시기 바랍니다.!!!!,**
- **!!! Run your solution code with the provided test case above and check whether it works properly !!!**

<Issue>

- 코드 작성시 주석을 적어주시기 바랍니다. 주석이 없는경우 **Cheating**으로 간주될 수 있습니다.
- 제공된 **testcase**는 채점 **case**에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
- 파일 입출력은 **argv[]** 를 사용하여 구현해주시기 바랍니다.
- 제출 마감 시간 이전의 가장 최신 버전의 **commit**을 기준으로 채점할 예정입니다.
- 제출 파일과, 폴더 **naming** 은 꼭 지정된 형식으로 해주셔야 합니다.
- **Please write down the detailed comments when writing the code. If there is no comment, it might be considered cheating.**
- **Provided test case is included in the test case for grading. Please check to see if it makes a proper result.**
- **Do not use a fixed file name when inputting and outputting files, but implement it using argv[] as in skeleton code.**
- **Scoring will be based on the latest version of commit before the deadline.**
- **The names of the .c file and directory should be named in proper format.**
- 출력시, 위 사진의 예시와 같은 형식으로 출력해주시면 됩니다. 모든 공백은 띄어쓰기 한칸입니다. 모든 출력 메시지의 알파벳은 소문자만 사용하여 출력합니다.
- -> 출력시 위 사진과 같이 커맨드에 알맞는 메시지 출력후 줄바꿈(\n).
- -> 출력파일 마지막줄의 경우, 줄바꿈(\n) 후 EOF.
- **All the messages must be printed out according to the appropriate format as shown**

- -> newline(\n) after each output message.
- -> newline(\n) after the last output message.

- 아래와 같이 git 프로젝트 폴더에 "lab05" 폴더 생성후, "lab05" 폴더 안에 "20XXXXXXXXX.c" 파일을 위치시키시면 됩니다.
- After creating the "lab05" directory in the git-project-directory as below, place the "20XXXXXXXXXX.c" file in the "lab05" directory.

[illegible]