

Lab 05 (3/31) Threaded Tree & Traversal

Due : ~2021.04.05.(Tue) 23:59, Late Submission : ~2021.04.06.(Thu) 16:59

- **Threaded Tree & Inorder Traversal Implementation**
- 아래의 사진과 같이 [input].txt 파일을 입력받아 Threaded Tree를 생성하고 편집하는 기능을 구현하고, Inorder Traversal의 결과 값을 [output].txt 파일에 출력하여 저장한다.
- **Implement the function of creating and editing a Threaded Tree by receiving the [input].txt file as input as below and the output result of Inorder Traversal is stored in the [output].txt file.**

```
열기(O)▼ 파일 input1.txt /data/data_bac... 저장(S) 열기(O)▼ 파일 output1.txt /data/data_bac... 저장(S)
8
1 2 3 4 5 6 7 8
inorder traversal : 8 4 2 5 1 6 3 7
```

일반 텍스트 ▼ 탭 너비: 4 ▼ 1행, 1열 ▼ 삽입 일반 텍스트 ▼ 탭 너비: 4 ▼ 1행, 1열 ▼ 삽입

<input>

- 첫째줄 : Threaded Tree에 insert할 Node의 개수 (100 이하의 자연수)
- 둘째줄 : Threaded Tree에 삽입할 Node들의 Key값 (10000 이하의 자연수, 공백 한칸씩을 사이에 두고 작성됨)
- first line : the number of nodes to be inserted into Threaded Tree (Natural Number less or equal to 100)
- second line : the key values of the nodes to be inserted into Threaded Tree (All the values are typed with single space between each other in a line, and the key value is a natural number less or equal to 10000)

<output>

- 생성된 Threaded Tree를 Inorder로 Traverse한 결과 출력.
- Inorder Traversal result of generated Threaded Tree.

<Structure & Function Format>

```
typedef struct ThreadedTree* ThreadedPtr;
typedef int ElementType;

struct ThreadedTree {
    int left_thread; // flag if ptr is thread
    ThreadedPtr left_child;
    ElementType data;
    ThreadedPtr right_child;
    int right_thread; // flag if ptr is thread
} ThreadedTree;
```

```
ThreadedPtr CreateTree();
int Insert(ThreadedPtr root, int root_idx, ElementType data, int idx);
void printInorder(ThreadedPtr root);
void DeleteTree(ThreadedPtr 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) 20XXXXXXXXXX.c

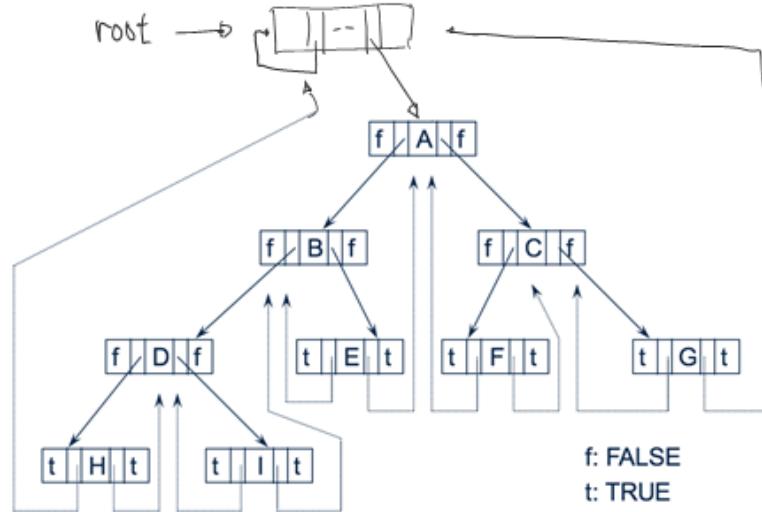
<Execution>

- gcc 20XXXXXXXXXX.c -o 20XXXXXXXXXX
- ./20XXXXXXXXXX [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 없는 경우 Cheating으로 간주될 수 있습니다.
- 제공된 testcase는 채점 case에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
- 파일 입출력은 `argv[]`를 사용하여 구현해주시기 바랍니다.
- 제출 마감 시간 이전의 가장 최신 버전의 commit을 기준으로 채점할 예정입니다.
- 제출 파일과, 폴더 naming은 꼭 지정된 형식으로 해주세요 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.
 - >출력파일 첫째줄의 마지막 숫자 출력후 EOF
- All the messages must be printed out according to the appropriate format as shown in the example above. All spaces are one space. Only lowercase letters should be used for the alphabets in output messages.
- >EOF right after the last key value in the line in the output file.
- Threaded Tree에 값들을 Insert 할 때는 입력파일의 왼쪽 숫자 부터 index 순서대로 입력해주세요 합니다.
- ex) **input : “A B C D E F G H I”** 일 때 아래의 사진과 같이 **Complete Binary Tree**의 형태로 Insert 하시면 됩니다.
- When inserting key values into the Tree, it should be inserted in the order of index from the left to right.

- ex) input : “A B C D E F G H I” -> Insertion should be done as **Complete Binary Tree** shape as below.



- 위 사진의 경우, root의 index는 0, A의 index는 1, B의 index는 2, C의 index는 3, D의 index는 4, E의 index는 5, F의 index는 6, G의 index는 7, H의 index는 8, I의 index는 9입니다.
 - In the threaded tree example above, index of root is 0, index of A is 1, index of B is 2, index of C is 3, index of D is 4, index of E is 5, index of F is 6, index of G is 7, index of H is 8, and the index of I is 9.
 - **printInorder()** 함수의 경우, **Recursive** 형태의 구현은 금지됩니다. (ex: **F()** 함수 내부에서 똑같은 **F()**함수를 호출하는 행위 금지.) - **Recursive** 형태 사용시 **0점**처리 됩니다.
 - **In the case of the function “printInorder()”, Implementing in a recursive form is prohibited. (ex: calling F() inside of F() is banned.) - 0 score for Recursive form.**

<Directory Format>

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

2022 CSE2010 20xxxxxxxxx/ (GitLab project directory)

10

---lab03/

-----2020XXXXXX.c

... .

---lab04/

-----2020XXXXXX.c

5

---lab05/
222

-----2020XXXXXX.0