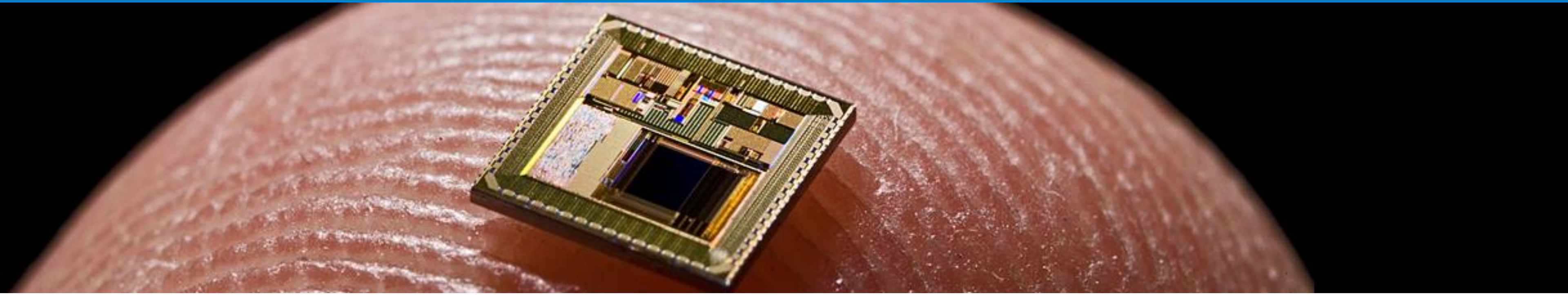
Data Structure

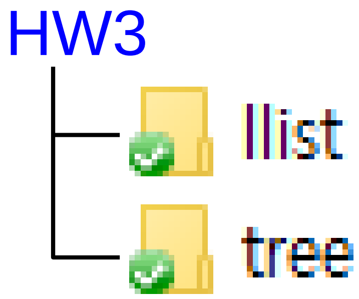
HW3

- BST implementation

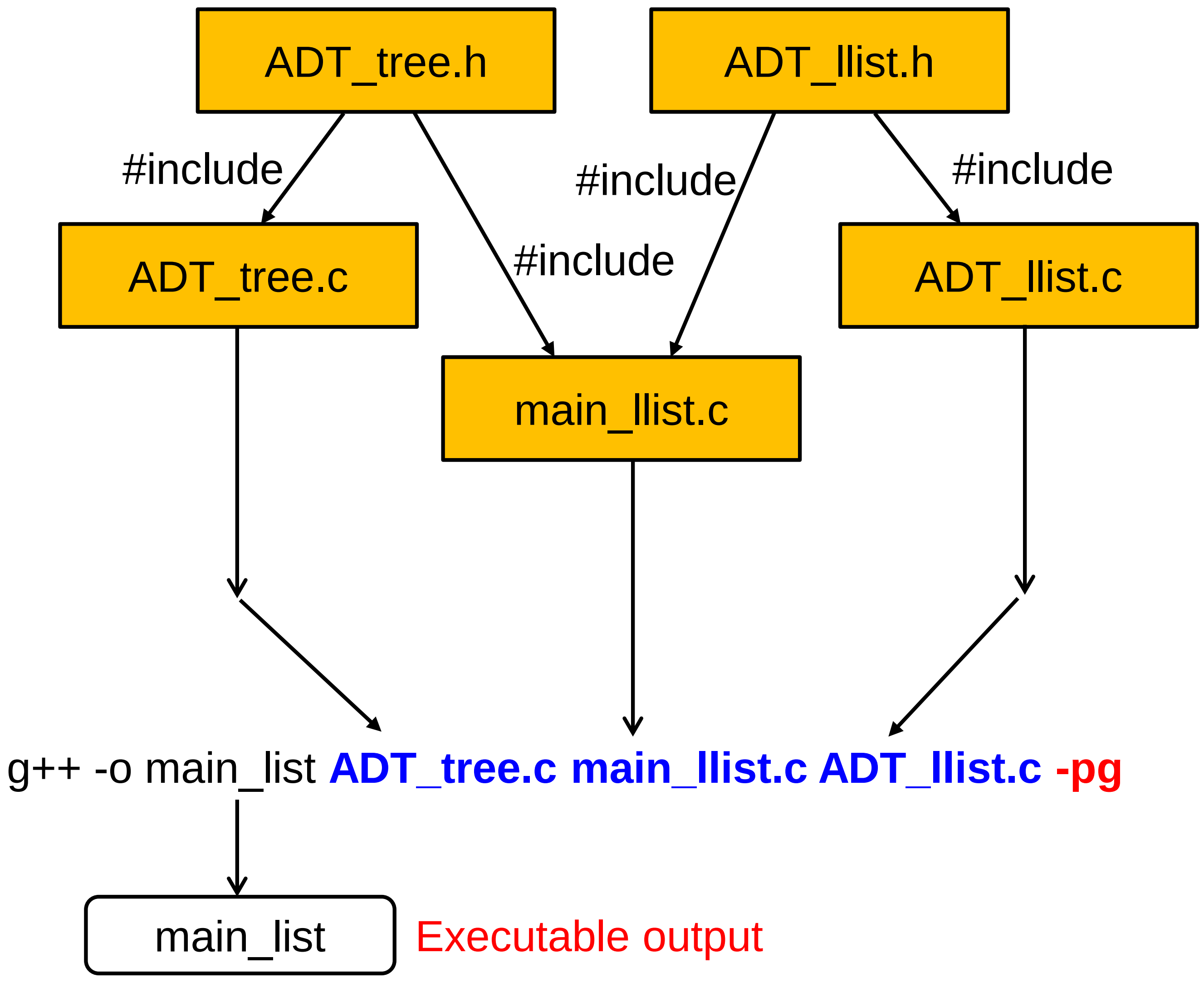


Prepare two blank folders

- Under homework top directory, for example, HW3



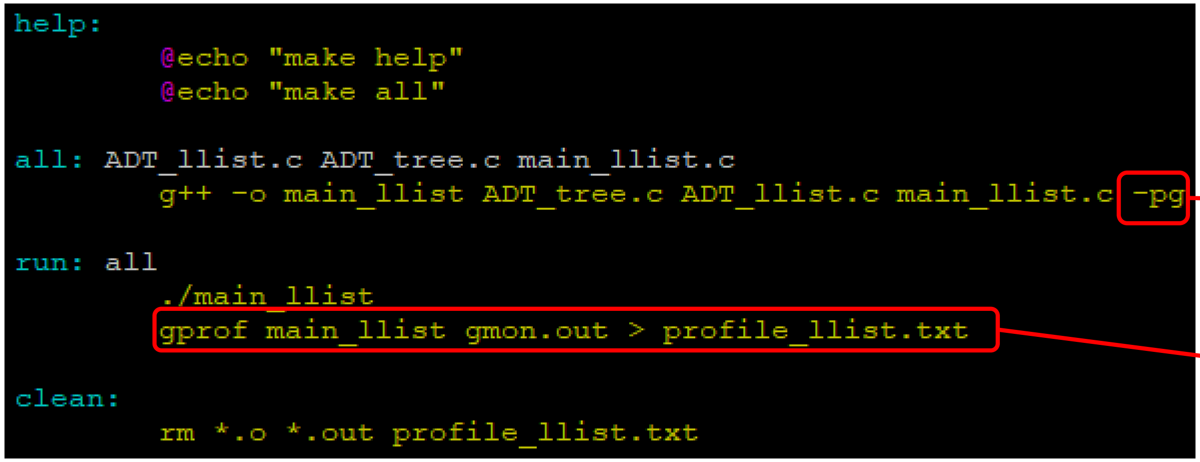
File Structure and Compilation Procedure (under folder ‘llist’)



```
g++ -o main_list ADT_tree.c main_llist.c ADT_llist.c -pg
```

Edit Makefile

Makefile

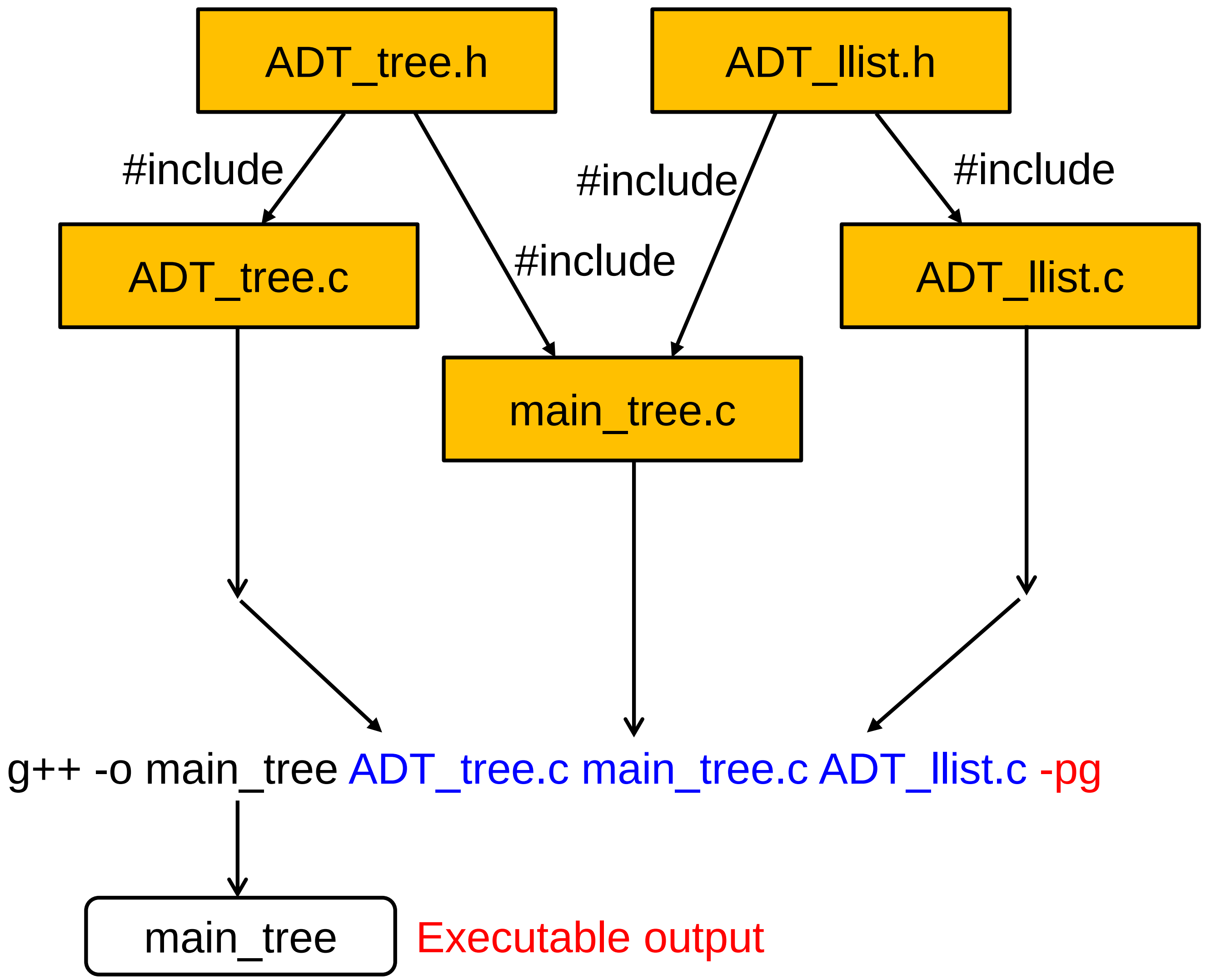


```
help:
        @echo "make help"
        @echo "make all"

all: ADT_llist.c ADT_tree.c main_llist.c
        g++ -o main_llist ADT_tree.c ADT_llist.c main_llist.c -pg

run: all
        ./main_llist
        gprof main_llist gmon.out > profile_llist.txt

clean:
        rm *.o *.out profile_llist.txt
```

Compile option for gprof

gprof command for profiling

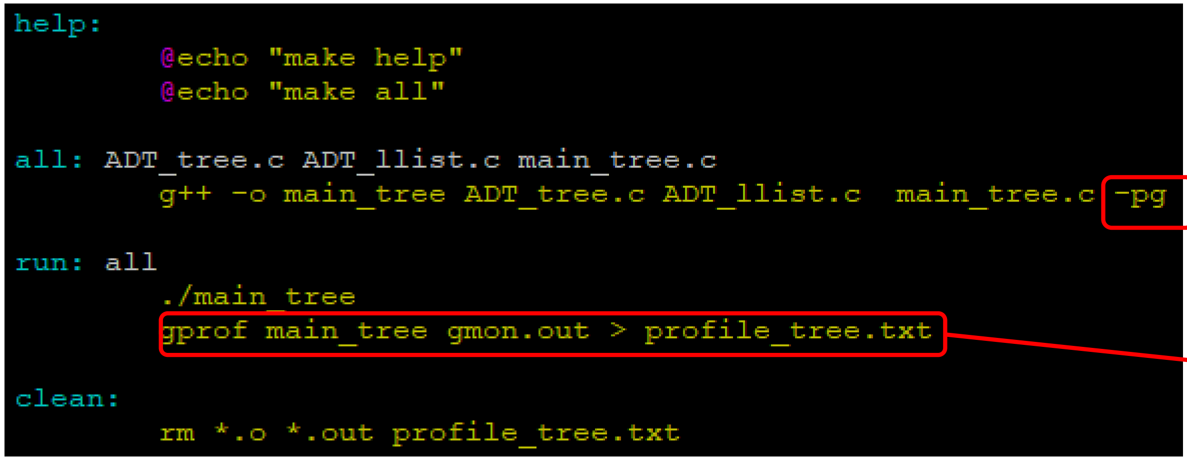
File Structure and Compilation Procedure (under folder ‘tree’)



```
g++ -o main_tree ADT_tree.c main_tree.c ADT_llist.c -pg
```

Edit Makefile

Makefile



```
help:
        @echo "make help"
        @echo "make all"

all: ADT_tree.c ADT_llist.c main_tree.c
        g++ -o main_tree ADT_tree.c ADT_llist.c  main_tree.c -pg

run: all
        ./main_tree
        gprof main_tree gmon.out > profile_tree.txt

clean:
        rm *.o *.out profile_tree.txt
```

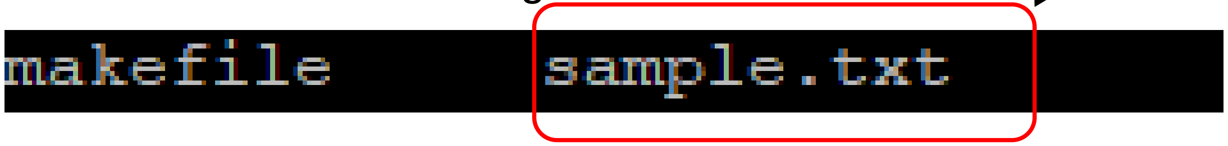
Compile option for gprof

gprof command for profiling

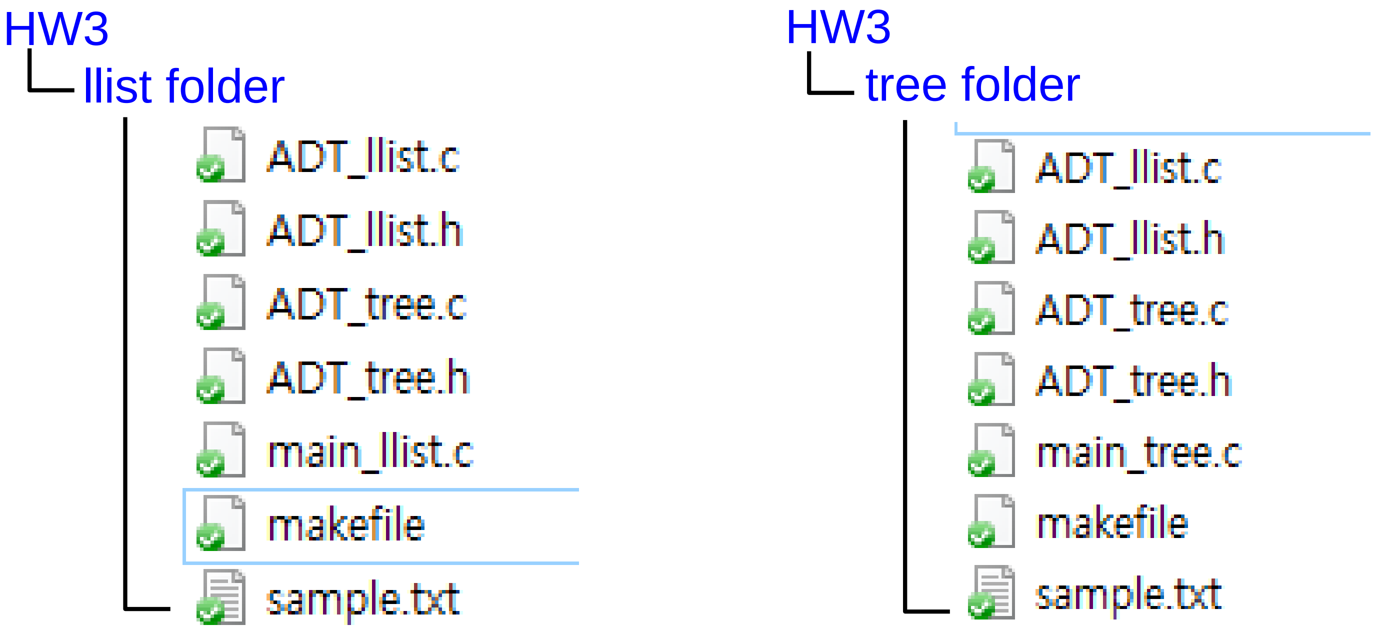
Download sample.txt

- Download sample.txt
- Extract into folder 'llist' and 'tree' under your homework top directory
  - After that, enter 'ls' command
    - You can see the following in terminal

Provided by website

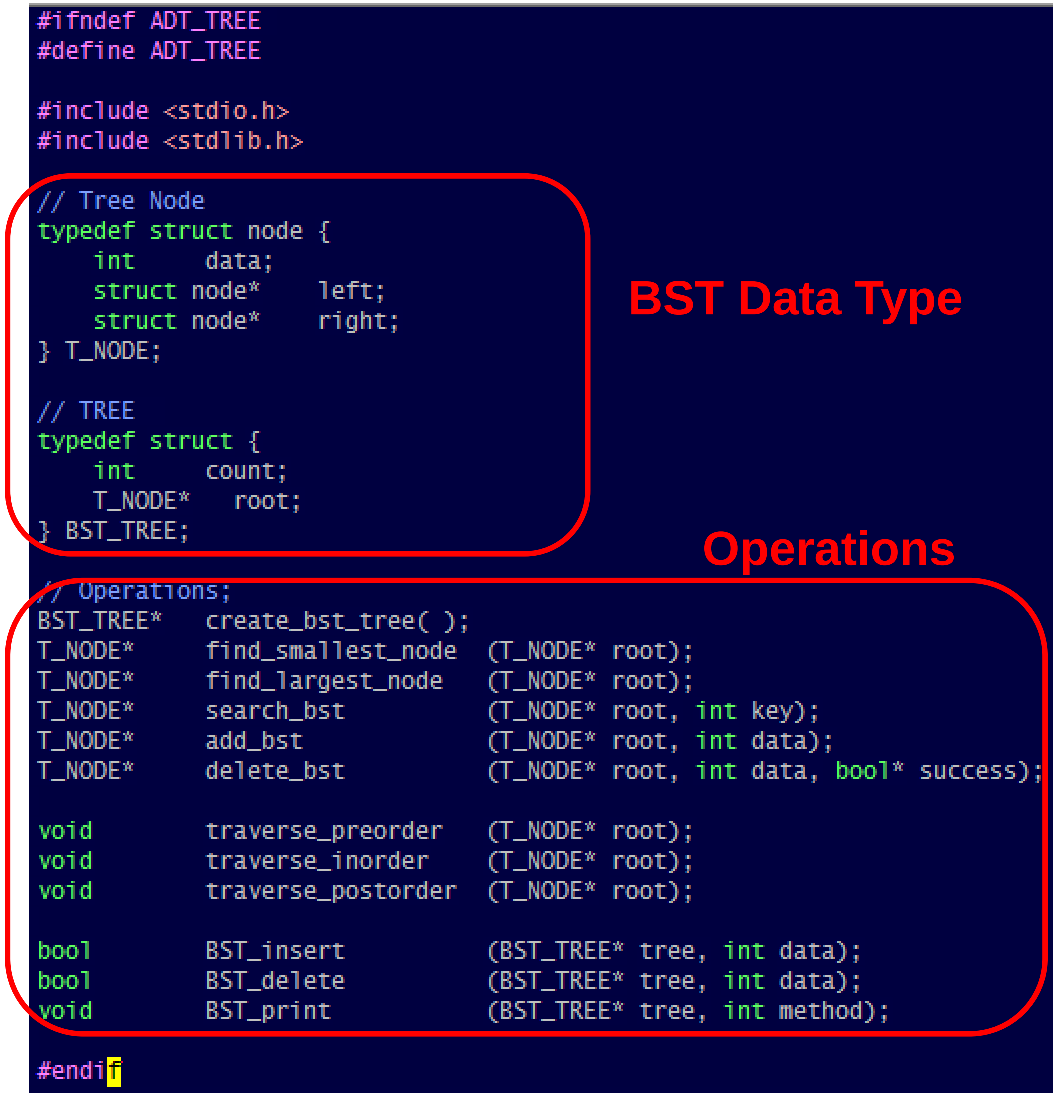


```
makefile          sample.txt
```



```
HW3
└─llist folder
    ADT_llist.c
    ADT_llist.h
    ADT_tree.c
    ADT_tree.h
    main_llist.c
    makefile
    sample.txt

HW3
└─tree folder
    ADT_llist.c
    ADT_llist.h
    ADT_tree.c
    ADT_tree.h
    main_tree.c
    makefile
    sample.txt
```

BST_TREE API

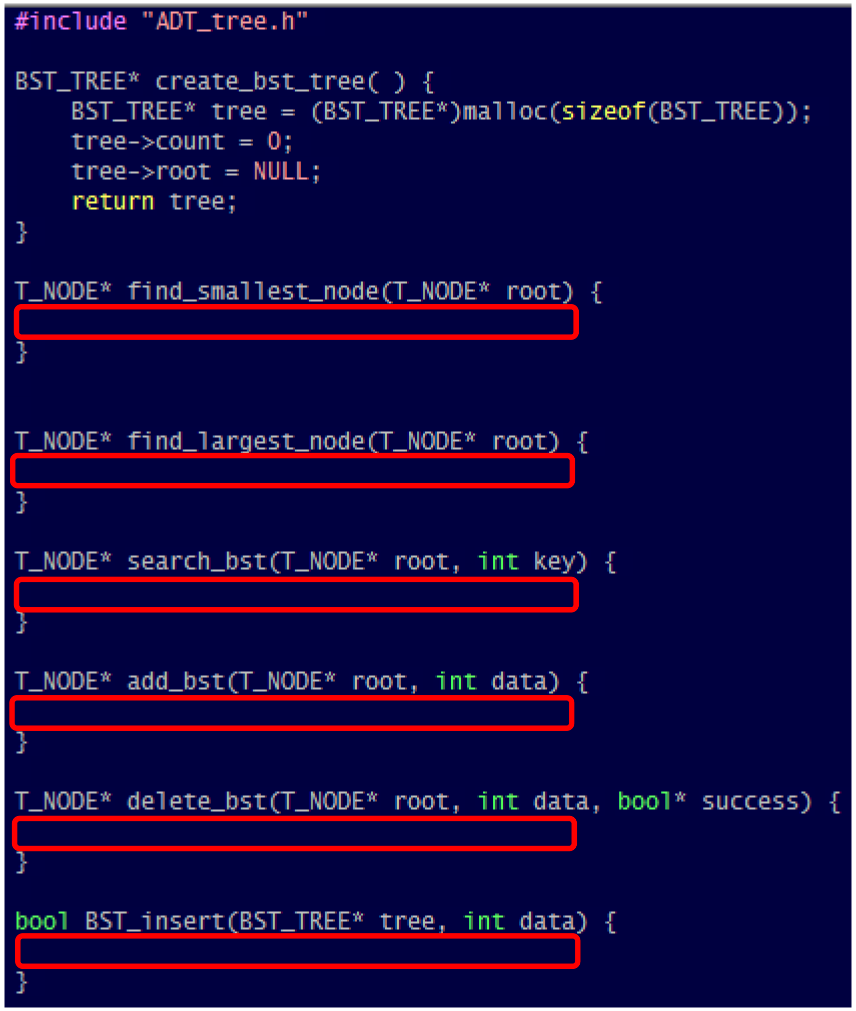


```c
#ifndef ADT_TREE
#define ADT_TREE

#include <stdio.h>
#include <stdlib.h>

// Tree Node
typedef struct node {
    int         data;
    struct node*    left;
    struct node*    right;
} T_NODE;

// TREE
typedef struct {
    int       count;
    T_NODE*    root;
} BST_TREE;

// Operations;
BST_TREE*    create_bst_tree( );
T_NODE*      find_smallest_node   (T_NODE* root);
T_NODE*      find_largest_node    (T_NODE* root);
T_NODE*      search_bst           (T_NODE* root, int key);
T_NODE*      add_bst              (T_NODE* root, int data);
T_NODE*      delete_bst           (T_NODE* root, int data, bool* success);

void         traverse_preorder    (T_NODE* root);
void         traverse_inorder     (T_NODE* root);
void         traverse_postorder   (T_NODE* root);

bool         BST_insert           (BST_TREE* tree, int data);
bool         BST_delete           (BST_TREE* tree, int data);
void         BST_print            (BST_TREE* tree, int method);

#endif
```

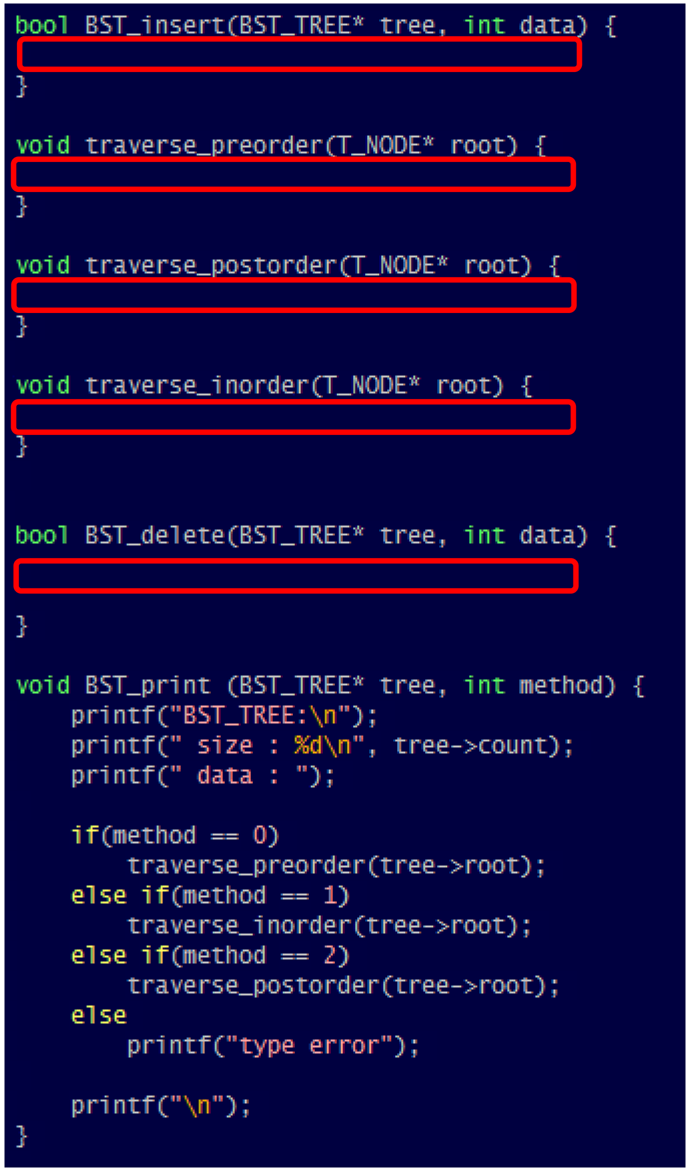
ADT_tree.h

BST_TREE API

ADT_tree.c



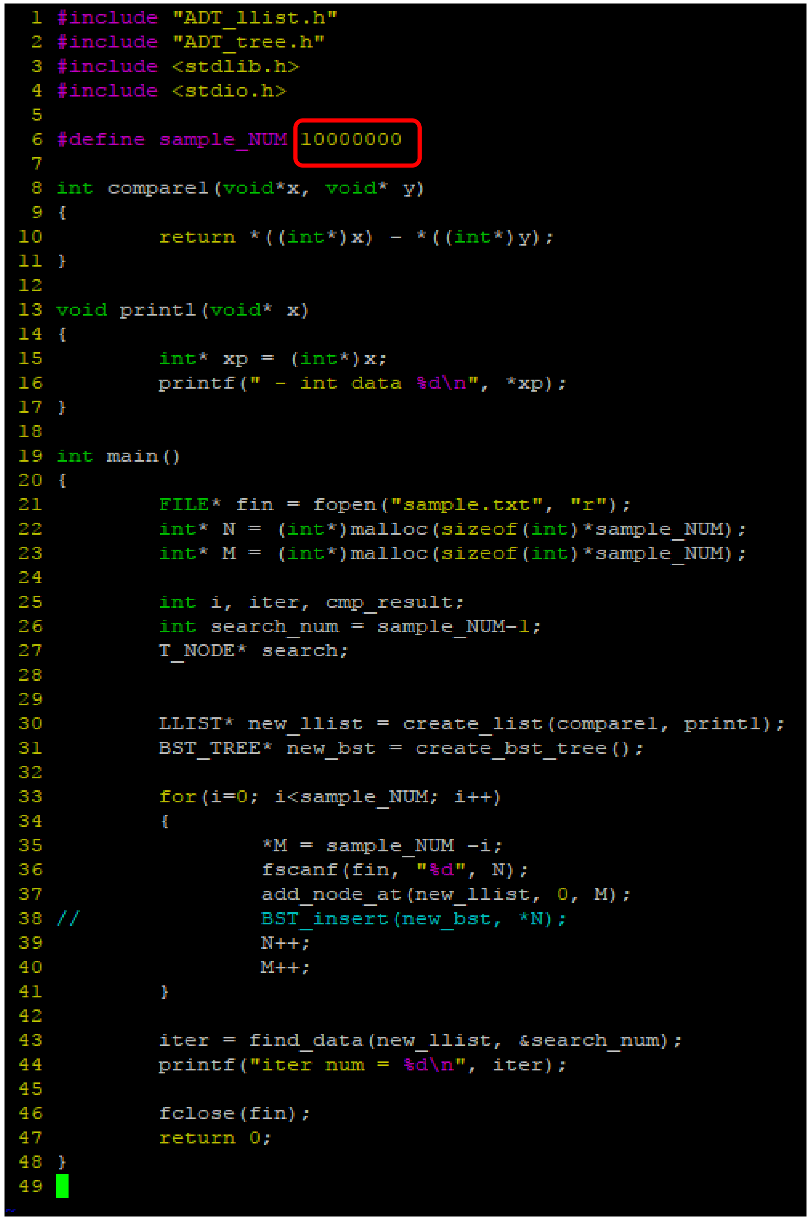
```
#include "ADT_tree.h"

BST_TREE* create_bst_tree( ) {
    BST_TREE* tree = (BST_TREE*)malloc(sizeof(BST_TREE));
    tree->count = 0;
    tree->root = NULL;
    return tree;
}

T_NODE* find_smallest_node(T_NODE* root) {

}


T_NODE* find_largest_node(T_NODE* root) {

}

T_NODE* search_bst(T_NODE* root, int key) {

}

T_NODE* add_bst(T_NODE* root, int data) {

}

T_NODE* delete_bst(T_NODE* root, int data, bool* success) {

}

bool BST_insert(BST_TREE* tree, int data) {

}
```



```
bool BST_insert(BST_TREE* tree, int data) {

}

void traverse_preorder(T_NODE* root) {

}

void traverse_postorder(T_NODE* root) {

}

void traverse_inorder(T_NODE* root) {

}


bool BST_delete(BST_TREE* tree, int data) {


}

void BST_print (BST_TREE* tree, int method) {
    printf("BST_TREE:\n");
    printf(" size : %d\n", tree->count);
    printf(" data : ");

    if(method == 0)
        traverse_preorder(tree->root);
    else if(method == 1)
        traverse_inorder(tree->root);
    else if(method == 2)
        traverse_postorder(tree->root);
    else
        printf("type error");

    printf("\n");
}
```

9

Insert into LLIST and search one of them



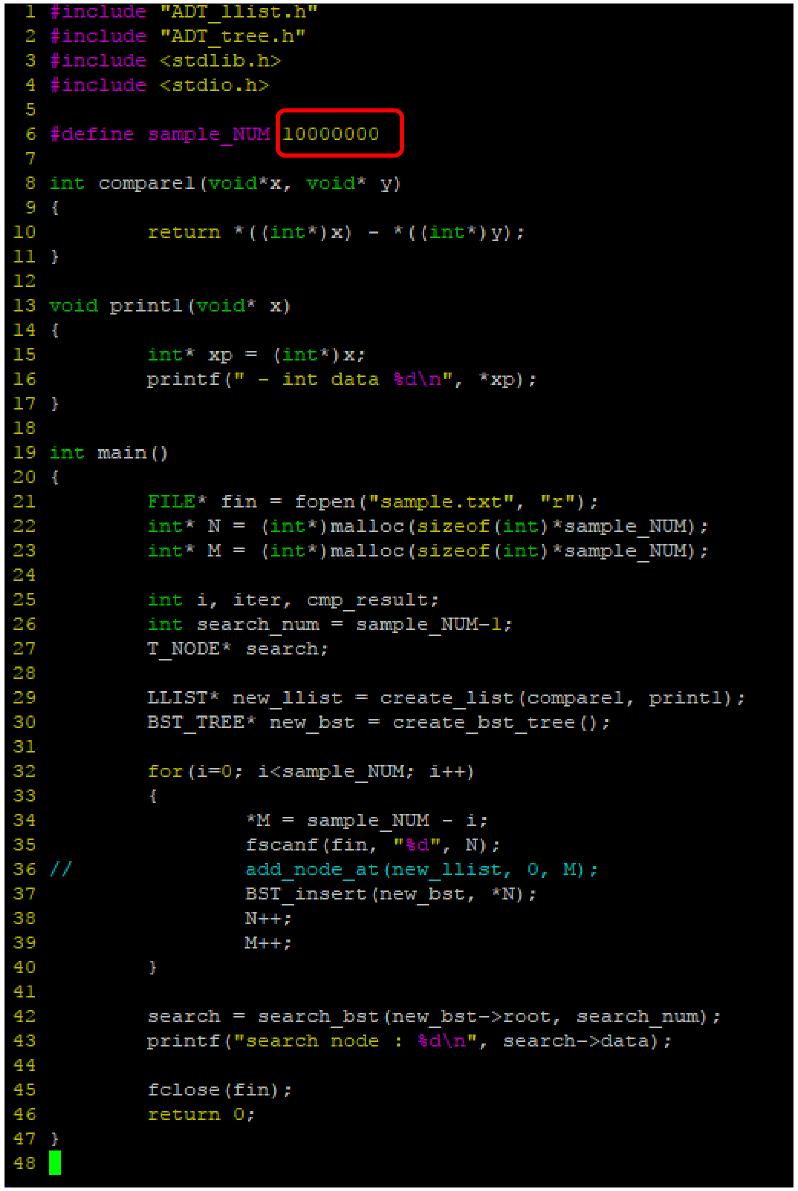
```c
#include "ADT_llist.h"
#include "ADT_tree.h"
#include <stdlib.h>
#include <stdio.h>

#define sample_NUM 10000000

int compare1(void*x, void* y)
{
        return *((int*)x) - *((int*)y);
}

void print1(void* x)
{
        int* xp = (int*)x;
        printf(" - int data %d\n", *xp);
}

int main()
{
        FILE* fin = fopen("sample.txt", "r");
        int* N = (int*)malloc(sizeof(int)*sample_NUM);
        int* M = (int*)malloc(sizeof(int)*sample_NUM);

        int i, iter, cmp_result;
        int search_num = sample_NUM-1;
        T_NODE* search;


        LLIST* new_llist = create_list(compare1, print1);
        BST_TREE* new_bst = create_bst_tree();

        for(i=0; i<sample_NUM; i++)
        {
                *M = sample_NUM -i;
                fscanf(fin, "%d", N);
                add_node_at(new_llist, 0, M);
//              BST_insert(new_bst, *N);
                N++;
                M++;
        }

        iter = find_data(new_llist, &search_num);
        printf("iter num = %d\n", iter);

        fclose(fin);
        return 0;
}
```

main_llist.c

- Add sample data to LLIST and BST
- Search the last data in LLIST

10

Insert into BST and search one of them



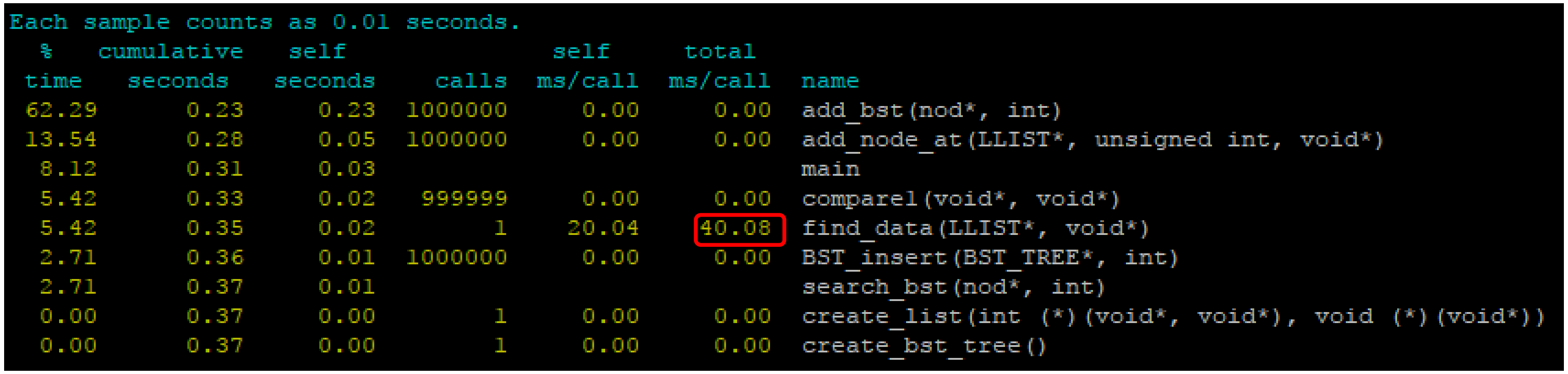
```
#include "ADT_llist.h"
#include "ADT_tree.h"
#include <stdlib.h>
#include <stdio.h>

#define sample_NUM 10000000

int compare1(void*x, void* y)
{
        return *((int*)x) - *((int*)y);
}

void print1(void* x)
{
        int* xp = (int*)x;
        printf(" - int data %d\n", *xp);
}

int main()
{
        FILE* fin = fopen("sample.txt", "r");
        int* N = (int*)malloc(sizeof(int)*sample_NUM);
        int* M = (int*)malloc(sizeof(int)*sample_NUM);

        int i, iter, cmp_result;
        int search_num = sample_NUM-1;
        T_NODE* search;

        LLIST* new_llist = create_list(compare1, print1);
        BST_TREE* new_bst = create_bst_tree();

        for(i=0; i<sample_NUM; i++)
        {
                *M = sample_NUM - i;
                fscanf(fin, "%d", N);
//              add_node_at(new_llist, 0, M);
                BST_insert(new_bst, *N);
                N++;
                M++;
        }

        search = search_bst(new_bst->root, search_num);
        printf("search node : %d\n", search->data);

        fclose(fin);
        return 0;
}

```

main_tree.c

- **Add sample data to LLIST and TREE**
- **Search the data in TREE**

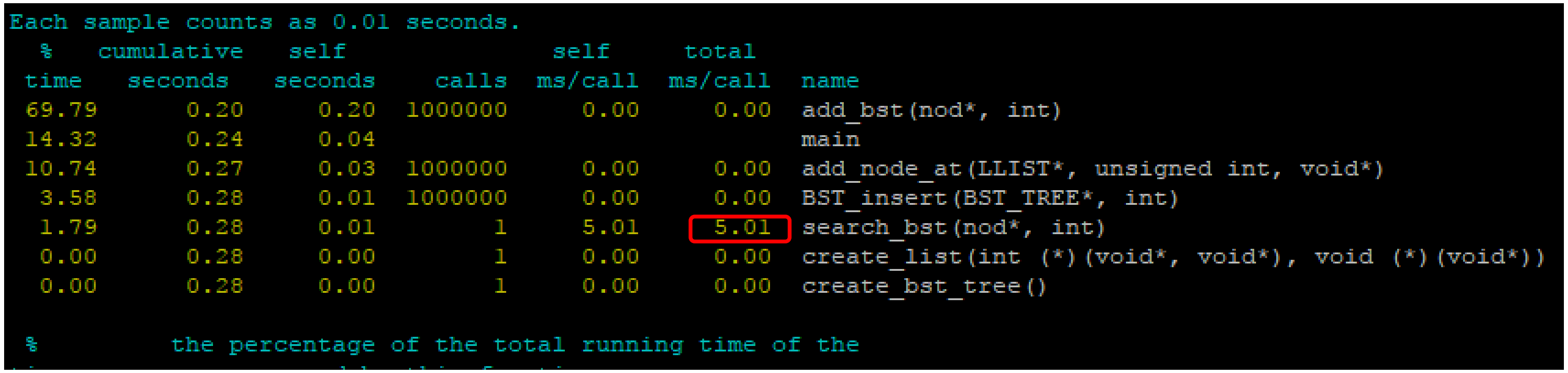
Profiling Result

Run ‘main_list’ and open ‘profile_list.txt’ under llist folder



```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 62.29      0.23     0.23  1000000     0.00     0.00  add_bst(nod*, int)
 13.54      0.28     0.05  1000000     0.00     0.00  add_node_at(LLIST*, unsigned int, void*)
  8.12      0.31     0.03                             main
  5.42      0.33     0.02   999999     0.00     0.00  compare1(void*, void*)
  5.42      0.35     0.02        1    20.04    40.08  find_data(LLIST*, void*)
  2.71      0.36     0.01  1000000     0.00     0.00  BST_insert(BST_TREE*, int)
  2.71      0.37     0.01                             search_bst(nod*, int)
  0.00      0.37     0.00        1     0.00     0.00  create_list(int (*)(void*, void*), void (*)(void*))
  0.00      0.37     0.00        1     0.00     0.00  create_bst_tree()
```

Run ‘main_tree’ and open ‘profile_tree.txt’ under tree folder



```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 69.79      0.20     0.20  1000000     0.00     0.00  add_bst(nod*, int)
 14.32      0.24     0.04                             main
 10.74      0.27     0.03  1000000     0.00     0.00  add_node_at(LLIST*, unsigned int, void*)
  3.58      0.28     0.01  1000000     0.00     0.00  BST_insert(BST_TREE*, int)
  1.79      0.28     0.01        1     5.01     5.01  search_bst(nod*, int)
  0.00      0.28     0.00        1     0.00     0.00  create_list(int (*)(void*, void*), void (*)(void*))
  0.00      0.28     0.00        1     0.00     0.00  create_bst_tree()

 %         the percentage of the total running time of the
```

- You can see the difference between LLIST and TREE.

Submit (Due: 11/28, PM 10:00)

- Korean student needs to submit two files, into ABEEK website
  - (1) Source code:
    - Compress your homework folder, named hw3_[id].zip
      - For example, **hw3_20161235.zip**
  - (2) Report
    - In addition, **attach the report** (Microsoft word format) to explain your homework in terms of implementation.
- Foreign students have to mail me directly with these two files as attachment
  - boltanut@knu.ac.kr

Kyungpook National University /
Daejin Park

Cloud-Connected IoT System Platform Lab.
http://CCIoTLab.com/come331

To be continued …