資料結構緒論 作業四

TA：
陳以新：n26120812@gs.ncku.edu.tw
吳定洋：n26120838@gs.ncku.edu.tw
徐子桓：n26120888@gs.ncku.edu.tw

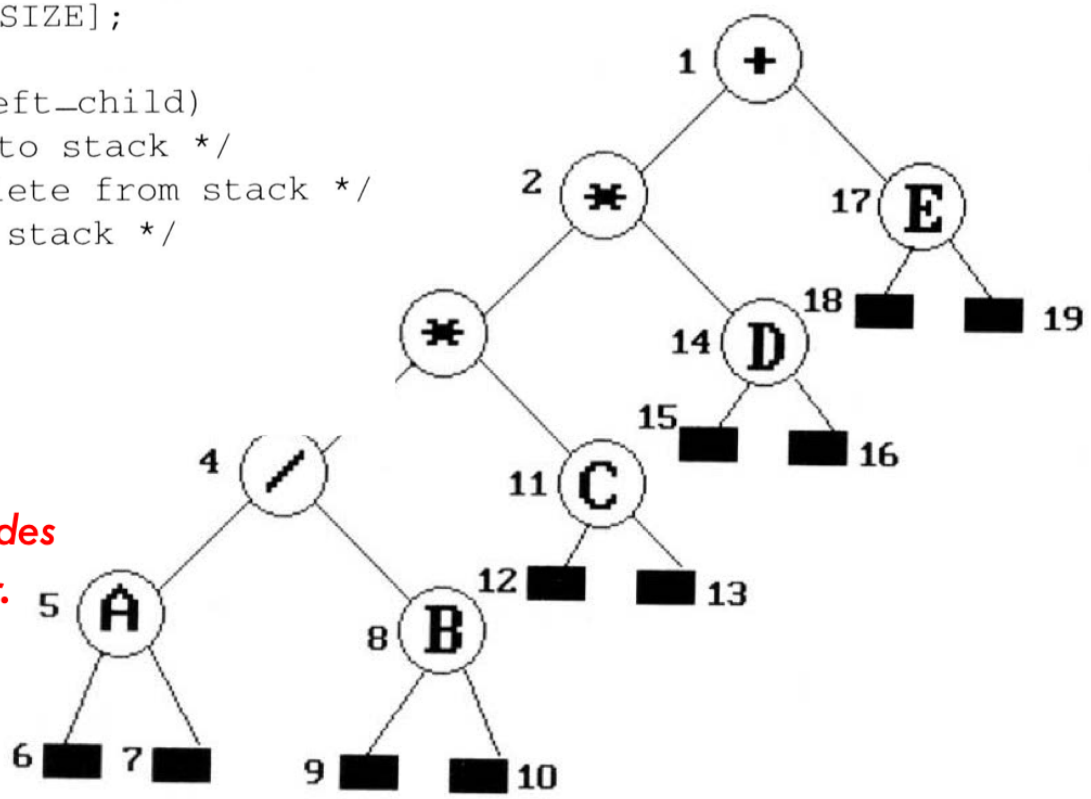
Homework 4

- In this homework, you should construct 3 binary trees as shown in the slides, and use the stack created in previous homework (DON'T use the stack given by STL) to implement Inorder Traversal with the iterative method (DON'T use recursion).
- Write your code in the given file (main.c & buildTree.c)
- Given File
  1. main.c (implement stack & inorder traversal)
  2. buildTree.c (construct 3 binary trees)
  3. buildTree.h (should be included in main.c & buildTree.c)

Pseudo code

```
void iter_inorder(tree_pointer node)
{
   int top = -1; /* initialize stack */
   tree_pointer stack[MAX_STACK_SIZE];
   for (;;) {
      for(; node; node = node->left_child)
         add(&top, node); /* add to stack */
      node = delete(&top); /* delete from stack */
      if (!node) break; /* empty stack */
      printf("%d", node->data);
      node = node->right_child;
   }
}
```

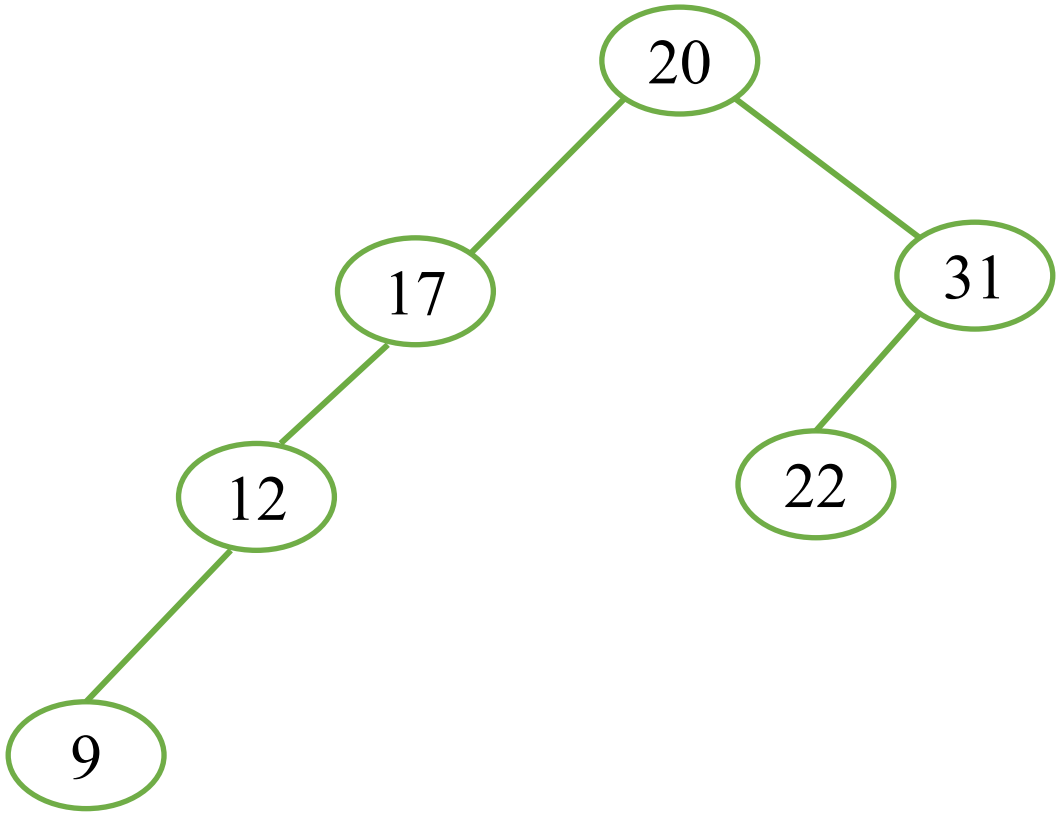
The equivalent iterative functions simulate the recursion.
We adds nodes to and remove nodes from our stack in the same manner.



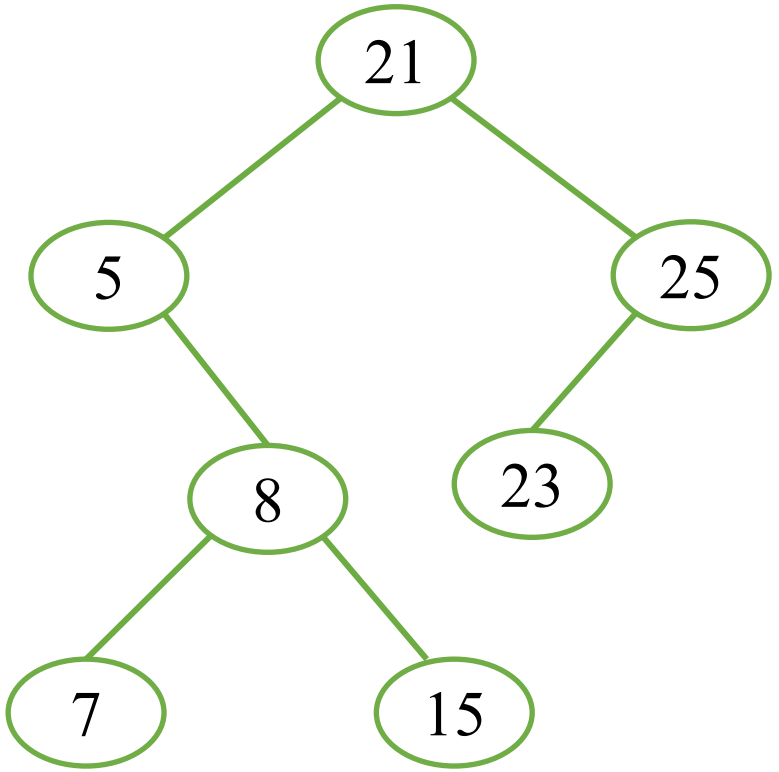
output: A /B *C *D +E

27

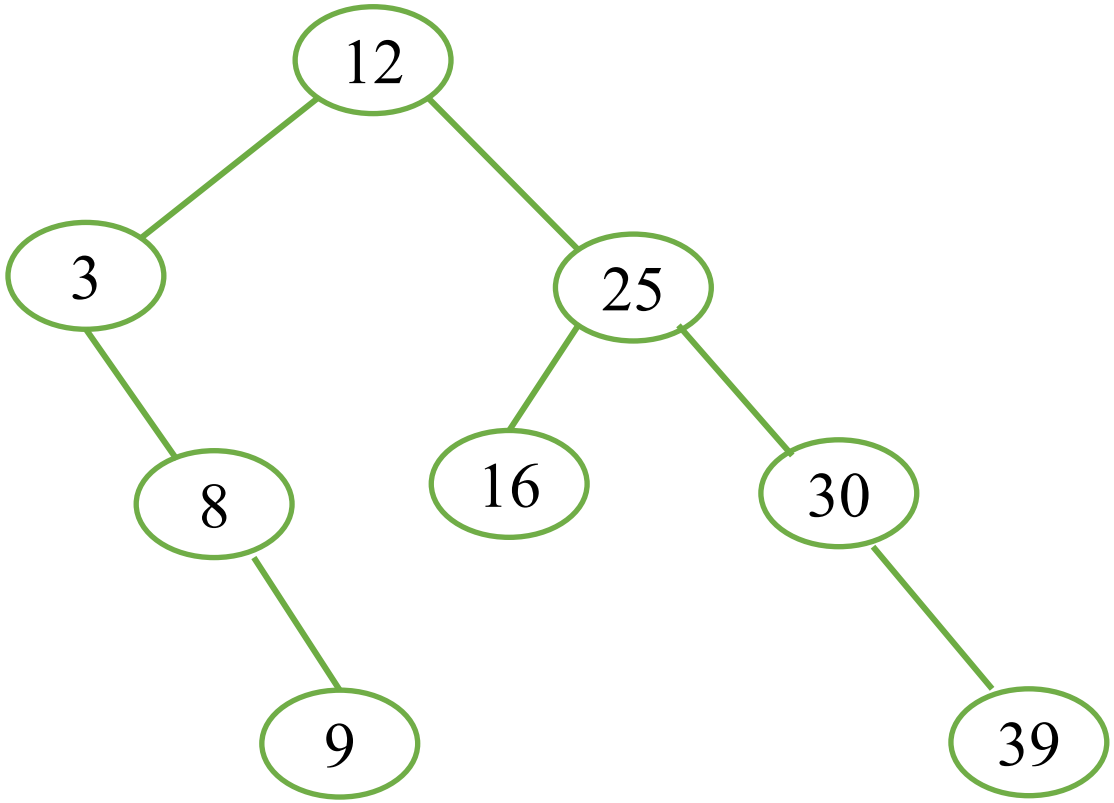
Tree 1



Tree 2



Tree 3



Function Explanation

- This section is in “buildTree.c”
- You should construct the above 3 trees in these 3 functions.

```
treeNode *buildTree1()
{
    /* Build Tree 1 in this function and return the root. */
}
treeNode *buildTree2()
{
    /* Build Tree 2 in this function and return the root. */
}
treeNode *buildTree3()
{
    /* Build Tree 3 in this function and return the root. */
}
```

Function Explanation

- This function is in “buildTree.c”
- This function helps you add a new node to a binary tree. It’s not a mandatory part in this homework. Whether to use it or not is up to you.

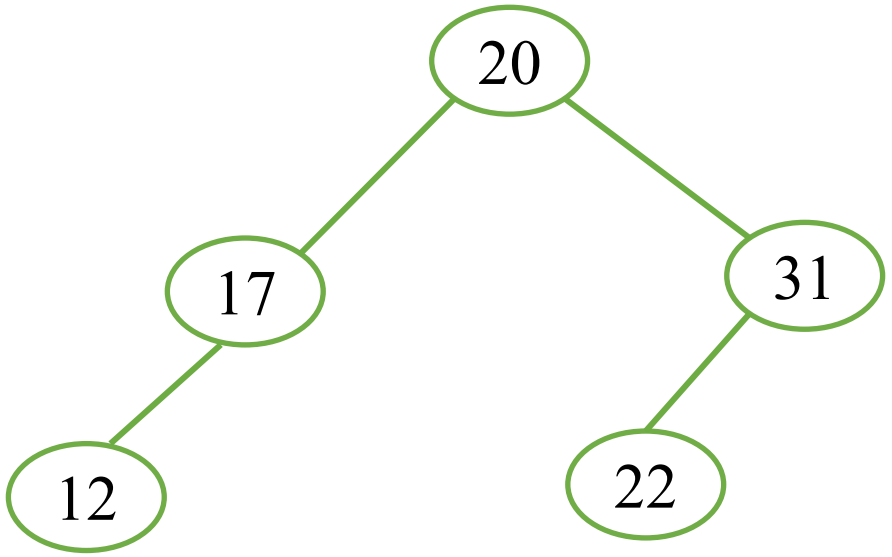
```c
treeNode *newNode(int data)
{
    /* This function helps you add a new node to a binary tree. */
    treeNode *node = (treeNode *)malloc(sizeof(treeNode));
    node->val = data;
    node->left = node->right = NULL;
    return node;
}
```

Function Explanation

- This function is in “main.c”
- You should implement inorder traversal and print your answer in this function (please follow the format on next page).
- Please implement your STACK in “main.c”

```
void iter_inorder(treeNode *node)
{
    /* Implement inorder traversal and print the result in this functon.*/
}
```

Output Format (Example)



In this example, your output should be:
12 17 20 22 31
Please follow the format to print your answer.

```
elsiepro@ElsieMacBook 資結助教 % cd "/Users/elsie96124_1
sers/elsie96124_1/Desktop/112-2/資結助教/"myhw4
Inorder Traversal of Tree 1:
12 17 20 22 31
Inorder Traversal of Tree 2:
4 2 5 1 3
Inorder Traversal of Tree 3:
4 2 5 1 3 %
```

This picture is just an example for your reference.

Requirement

- Implement 4 functions in the given file.
  1. `treeNode *buildTree1();`
  2. `treeNode *buildTree2();`
  3. `treeNode *buildTree3();`
  4. `void iter_inorder(treeNode *node);`
- Use the stack created by yourself in “main.c”. (You can use the stack created in previous homework, but you should NOT use that given by STL.)
- Implement iterated inorder traversal. (You should NOT use recursion here.)
- Please follow the output format shown on the above slide.
- Do NOT edit code in main function and the struct treeNode.

NOTE

- This section is in “main.c”
- You should NOT edit this section (main function).

```c
int main()
{
    /* You should not edit the code here. */
    printf("Inorder Traversal of Tree 1:\n");
    treeNode *tree1 = buildTree1();
    iter_inorder(tree1);
    printf("\nInorder Traversal of Tree 2:\n");
    treeNode *tree2 = buildTree2();
    iter_inorder(tree2);
    printf("\nInorder Traversal of Tree 3:\n");
    treeNode *tree3 = buildTree3();
    iter_inorder(tree3);
    return 0;
}
```

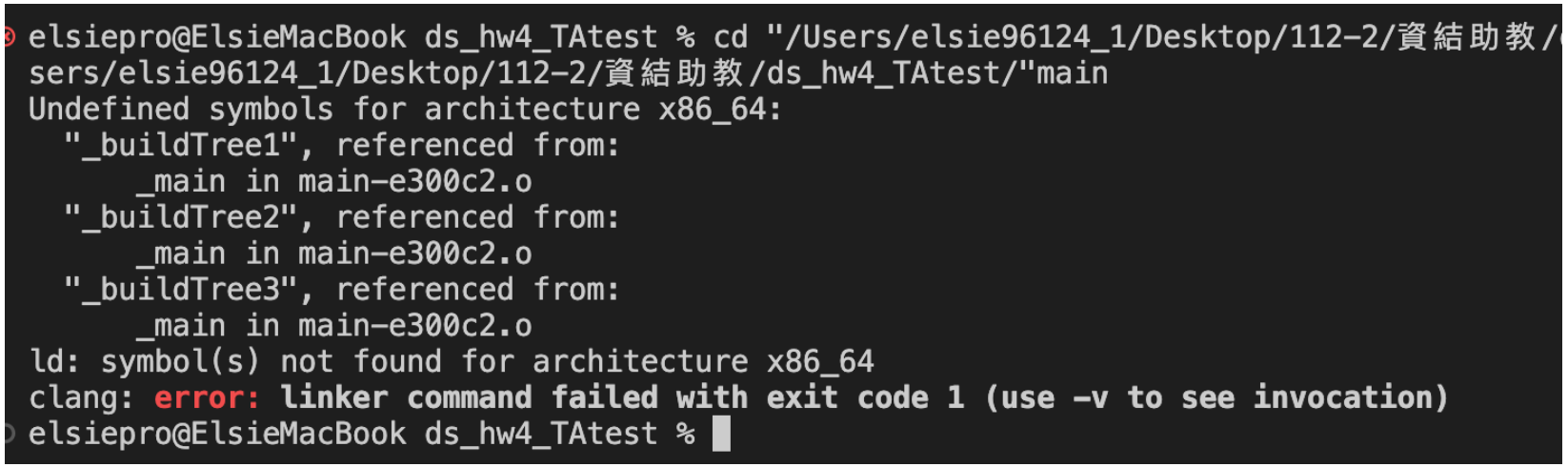
NOTE

- This section is in “buildTree.h”.
- You should NOT edit this struct.
- You MUST use this struct when you construct a binary tree.

```c
typedef struct treeNode
{
    /* You MUST use this struct when you build a binary tree. */
    /* You should not edit the code here. */
    int val;
    struct treeNode *left;
    struct treeNode *right;
} treeNode;
```

NOTE

- If you add another function in “buildTree.(c/cpp)” , please declare its Function Prototype in “buildTree.h”.
- If you compile and run main.(c/cpp) directly, there might be an error shown in the picture. Please check which compiler you’re using, and follow the following steps to compile and run your code.



Compile & Run

Compile and run your code with command line

Compile with GCC (for C)

```
$ gcc -c buildTree.c
$ gcc -c main.c
$ gcc -o main buildTree.o  main.o
$ ./main
```

Compile with Clang (for C/C++)

- For C:
  - $ clang -c buildTree.c
  - $ clang -c main.c
  - $ clang -o main buildTree.o main.o
  - $ ./main
- For C++:
  - $ clang -c buildTree.cpp
  - $ clang -c main.cpp
  - $ clang -o main buildTree.o main.o
  - $ ./main

Compile with G++ (for C++)

```
$ g++ -c buildTree.cpp
$ g++ -c main.cpp
$ g++ -o main buildTree.o  main.o
$ ./main
```

Grading Policy

評分標準

- 程式碼（需有適當註解）(80%)
  1. Construct 3 binary trees (20%)
  2. Iterative inorder traversal without recursion (45%)
  3. Stack created by yourself（在操作stack pop, push 的地方請務必在程式碼後面註解 // stack pop 、 // stack push）(15%)
- 報告書 (pdf) (20%)
  1. 程式碼概念說明(使用的資料結構、程式邏輯、演算法、各函式的功能等等，本次作業著重在如何以不使用recursive的方式對二元樹做 inorder traversal）
  2. 輸出結果截圖
- 嚴禁抄襲,否則當次作業以0分計算

作業繳交

- 繳交檔案請參照下圖，務必將所有檔案壓縮成1個zip檔繳交
- 作業說明檔(pdf)須包含輸出結果截圖，程式碼限制用c/c++
- 繳交期限：依 moodle 公告為準 (遲交一個禮拜 -20%，超過兩個禮拜拒收)