

Introduction to Data Structures

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

Homework 3B

- 40 points for coding evaluation
 - Submission format
 - File name: yourid_HW3B.c
 - Example: 2000123456_HW3B.c
 - File type: .c (NOT .cpp)
 - Submission site: <https://icampus.skku.edu>
 - Week 9: [Homework] 3B (code)

- 1 point for report
 - The report is not evaluated in detail but evaluated as Pass/Fail
 - Template: Homework Report Template.docx
 - Submission format
 - File name: yourid_HW3B.pdf
 - Example: 2000123456_HW3B.pdf
 - File type: .pdf
 - Submission site: <https://icampus.skku.edu>
 - Week 9: [Homework] 3B (report)

- Due date
 - 11/10 23:59 (no late submission accepted)

Rules for homework

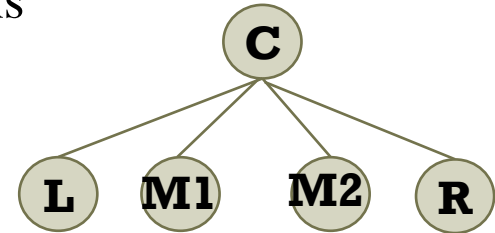
- You should follow instructions.
 - Compiler
 - You will get **no/less point** if your program cannot be complied with the specified compiler
 - Input/output format
 - You will get **no/less point** if TA's automatic evaluation program cannot parse your input or output.
 - Permitted modification scope
 - You will get **no/less point** if you modify code outside of the permitted modification scope
 - All other rules
 - You will get **severe penalty or no/less point** if you violate the given rules.

Compiler and input/output rules for homework

- Every implementation homework will be evaluated by TA's automatic evaluation program with the following compiler.
 - Compiler: GCC 7.X, 8.X, 9.X or 10.X
 - <https://gcc.gnu.org/>
 - You will get no/less point if your program cannot be compiled with GCC 7.X, 8.X, 9.X or 10.X.
 - For example, do not rely on visual studio.
 - You can use standard library such as *stdlib.h* and *math.h*.
- Input/output format
 - You will get no/less point if TA's automatic evaluation program cannot parse your input or output according to the following rules.
 - Use `stdin` and `stdout`

Problem

- Problem: QuadTree implementation using linked list
 - QuadTreeNode consists of left, middle1, middle2 and right tree.
 - Implement QuadTreeNode and its basic operations
 - Implement three traversal functions
 - preorder: C L M1 M2 R
 - inorder: L M1 C M2 R
 - postorder: L M1 M2 R C
 - Implement a function that counts the number of nodes



Output

■ Output

1: 2 3 4 5

2: 6 7 null null

3: 8 null null null

4: null null null null

5: null null null 9

6: null null null null

7: null null null null

8: null null null null

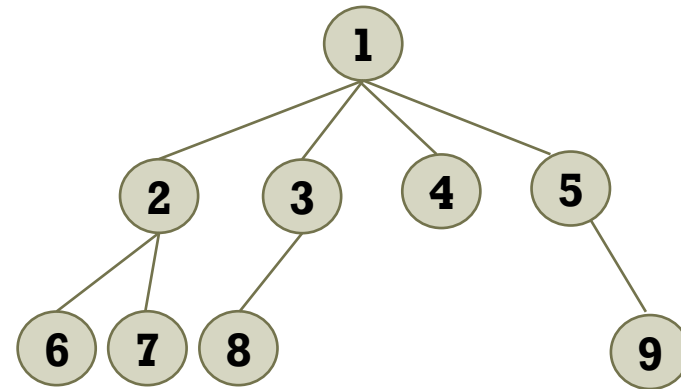
9: null null null null

1 2 6 7 3 8 4 5 9 -----> Preorder

6 7 2 8 3 1 4 5 9 -----> Inorder

6 7 2 8 3 4 9 5 1 -----> Postorder

9 -----> CountNodes



Template

■ You have a template.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* Modify from here */

/* Modify to here */

typedef int QuadData;

typedef struct _QuadTreeNode {
    QuadData item;
    struct _QuadTreeNode *left_child;
    struct _QuadTreeNode *middle1_child;
    struct _QuadTreeNode *middle2_child;
    struct _QuadTreeNode *right_child;
} QuadTreeNode;

void PrintTreeNode(QuadTreeNode *node);

QuadTreeNode *CreateNode(QuadData item);
void DestroyNode(QuadTreeNode *node);
void CreateLeftSubtree(QuadTreeNode *root, QuadTreeNode *left);
void CreateMiddle1Subtree(QuadTreeNode *root, QuadTreeNode *middle1);
void CreateMiddle2Subtree(QuadTreeNode *root, QuadTreeNode *middle2);
void CreateRightSubtree(QuadTreeNode *root, QuadTreeNode *right);
void preorder(QuadTreeNode *root);
void inorder(QuadTreeNode *root);
void postorder(QuadTreeNode *root);
int countNodes(QuadTreeNode *root);
QuadTreeNode *leftMost(QuadTreeNode* node);
/* Modify from here */

/* Modify to here */
```

```
int main() {
    QuadTreeNode *node1 = CreateNode(1);
    QuadTreeNode *node2 = CreateNode(2);
    QuadTreeNode *node3 = CreateNode(3);
    QuadTreeNode *node4 = CreateNode(4);
    QuadTreeNode *node5 = CreateNode(5);
    QuadTreeNode *node6 = CreateNode(6);
    QuadTreeNode *node7 = CreateNode(7);
    QuadTreeNode *node8 = CreateNode(8);
    QuadTreeNode *node9 = CreateNode(9);

    PrintTreeNode(node1);
    PrintTreeNode(node2);
    PrintTreeNode(node3);
    PrintTreeNode(node4);
    PrintTreeNode(node5);
    PrintTreeNode(node6);
    PrintTreeNode(node7);
    PrintTreeNode(node8);
    PrintTreeNode(node9);

    preorder(node1);
    printf("\n");
    inorder(node1);
    printf("\n");
    postorder(node1);
    printf("\n");
    printf("%d\n", countNodes(node1));

    DestroyNode(node1);
    DestroyNode(node2);
    DestroyNode(node3);
    DestroyNode(node4);
    DestroyNode(node5);
    DestroyNode(node6);
    DestroyNode(node7);
    DestroyNode(node8);
    DestroyNode(node9);

    return 0;
}
```

```
CreateLeftSubtree(node1, node2);
CreateMiddle1Subtree(node1, node3);
CreateMiddle2Subtree(node1, node4);
CreateRightSubtree(node1, node5);
```

```
CreateLeftSubtree(node2, node6);
CreateMiddle1Subtree(node2, node7);
```

```
CreateLeftSubtree(node3, node8);
CreateRightSubtree(node5, node9);
```

```
PrintTreeNode(node1);
PrintTreeNode(node2);
PrintTreeNode(node3);
PrintTreeNode(node4);
PrintTreeNode(node5);
PrintTreeNode(node6);
PrintTreeNode(node7);
PrintTreeNode(node8);
PrintTreeNode(node9);
```

```
preorder(node1);
printf("\n");
inorder(node1);
printf("\n");
postorder(node1);
printf("\n");
printf("%d\n", countNodes(node1));
```

```
DestroyNode(node1);
DestroyNode(node2);
DestroyNode(node3);
DestroyNode(node4);
DestroyNode(node5);
DestroyNode(node6);
DestroyNode(node7);
DestroyNode(node8);
DestroyNode(node9);
```

```
return 0;
```

Template

■ You have a template.

```
void PrintTreeNode(QuadTreeNode *node) {
    printf("%d: ", node->item);
    node->left_child == NULL ? printf("null ") : printf("%d ", node->left_child->item);
    node->middle1_child == NULL ? printf("null ") : printf("%d ", node->middle1_child->item);
    node->middle2_child == NULL ? printf("null ") : printf("%d ", node->middle2_child->item);
    node->right_child == NULL ? printf("null\n") : printf("%d\n", node->right_child->item);
}

QuadTreeNode *leftMost(QuadTreeNode* node) {
    if (node==NULL) return NULL;
    while (node->left_child != NULL || node->middle1_child != NULL) {
        if (node->left_child != NULL) node = node->left_child;
        else if (node->middle1_child != NULL) node = node->middle1_child;
    }
    return node;
}
```

```
/* Modify from here */
```

```
/* Modify to here */
```


Template

- You cannot modify the template except the space between `/*Modify from here*/` and `/*Modify to here*/`
- **Do not remove** `/*Modify from here*/` and `/*Modify to here*/`
- TA will copy the space and evaluate your code.
- You may add user-defined functions and header files between `/*Modify from here*/` and `/*Modify to here*/`.
- In the space, you need to implement the following functions. (Next page)

Template

- `QuadTreeNode *CreateNode(QuadData item);`
 - Create a `QuadTreeNode` with `item`
- `void DestroyNode(QuadTreeNode *node);`
 - Free the memory space of `node`
- `void CreateLeftSubtree(QuadTreeNode *root, QuadTreeNode *left);`
 - Attach the node of `left` as `left_child` of the node of `root`
- `void CreateMiddle1Subtree(QuadTreeNode *root, QuadTreeNode *middle1);`
 - Attach the node of `middle1` as `middle1_child` of the node of `root`
- `void CreateMiddle2Subtree(QuadTreeNode *root, QuadTreeNode *middle2);`
 - Attach the node of `middle2` as `middle2_child` of the node of `root`
- `void CreateRightSubtree(QuadTreeNode *root, QuadTreeNode *right);`
 - Attach the node of `right` as `right_child` of the node of `root`

Template

- `void preorder(QuadTreeNode *root);`
 - Traverse and print nodes using preorder starting from the root node
- `void inorder(QuadTreeNode *root);`
 - Traverse and print nodes using inorder starting from the root node
- `void postorder(QuadTreeNode *root);`
 - Traverse and print nodes using postorder starting from the root node
- `int countNodes(QuadTreeNode *root);`
 - Count the number node in the tree starting from the root node

Evaluation

■ Evaluation

- TA will test several cases by changing the main function.
- Read Pages 7~9 (regarding template) carefully.
- For each test case,
 - If your C code results in an answer within 10 seconds on a platform with average computing power,
 - If your output is perfect for four lines,
 - You get 100%.
 - Else if your output is valid for only for X lines ($1 \leq X \leq 3$)
 - You get $25\% * X$.
 - Else,
 - You get 0%.
 - Else,
 - You get 0%.