Practice #10

Trees (Binary Search Tree)

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

Practice #10 TO-DO List

To-Do	Submission	Notes
Binary Search Tree	Screenshot and source code (BST.h, BST.cpp)	p.65 ~ 77, Chapter 5

- Upload your screenshot and source codes on LMS by 11pm on 5/5 (Wed).
  - All your screenshots should be merged in one pdf file, screenshot.pdf.
  - Your pdf and all source codes should be compressed into zip file.
- File name: practice10_Your Student ID_Name.zip (only zip, not pdf, docx, c, etc)
  - ex) practice09_20400022_고윤민.zip

Binary Search Tree

- Implement a binary search tree class
  - Write BST.h and BST.cpp (BSTMain.cpp: no need to change)
    - BST.h: definition of BST class
    - BST.cpp: member functions of BST class
  - Refer to p.65 ~ 77, Chapter 5
  - You have to implement *delete_node()* and *print()* since our slides do not provide the source codes
    - *delete_node(int key)*: delete a node with *key*
    - *print()*: print all nodes' key using level order traversal
      - Queue class is necessary
      - Please refer to p.36, Chapter 5

Binary Search Tree

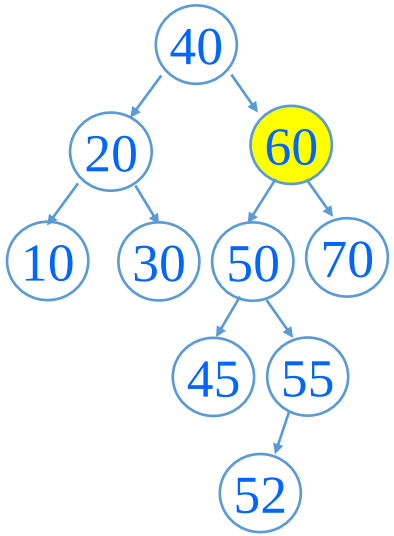
- Hint for *delete_node()* (recursive version)

```
void BST::delete_node(int key) {
    delete_node(root, key);
}

tree_node* BST::delete_node(tree_node *ptr, int key) {
    if (ptr == NULL)
        return NULL;

    if (ptr->key > key)
        ptr->left_child = delete_node(ptr->left_child, key);
    else if (ptr->key < key)
        ptr->right_child = delete_node(ptr->right_child, key);
    else {
        tree_node *temp_node;
        /*
        if ptr is a leaf node or a non-leaf node with a single child
            temp_node is a NULL if ptr is a leaf node or a single child if ptr is a non-leaf node with a child
            deallocate ptr
            return temp_node
        else if ptr is non-leaf node with two children
            temp_node is the largest element in ptr's left subtree or the smallest element in ptr's right subtree
            ptr's key is set to node's key
            delete the temp_node with key
        */
    }
    return ptr;
```

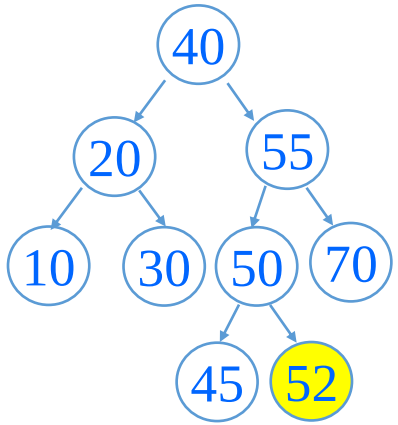
Binary Search Tree

- Expected results

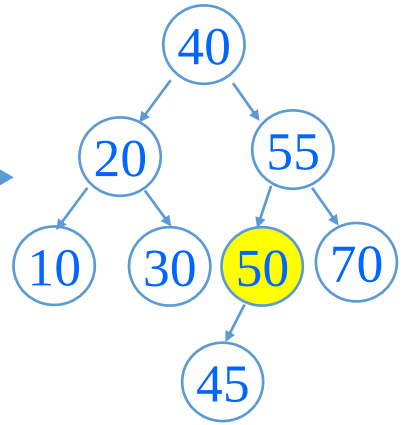
```
PS C:\ds\practice10\sol> .\BSTMain.exe
BST Print after Insert Node:
[40][20][60][10][30][50][70][45][55][52]

BST Search 20: Found
BST Search 80: Not Found

BST Print after Delete Node 60:
[40][20][55][10][30][50][70][45][52]

BST Print after Delete Node 52
[40][20][55][10][30][50][70][45]

BST Print after Delete Node 50
[40][20][55][10][30][45][70]
```



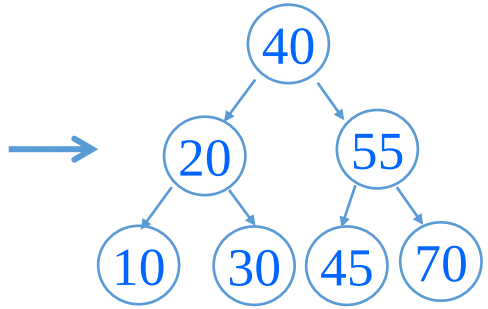
Deleting ‘60’



Deleting ‘52’



Deleting ‘50’



Deleting ‘50’