Lab09-Notes

In this lab you will be implementing a Binary Search Tree (BST). You will read from the input file and add the pokemons to the BST one by one (as nodes in the BST) and perform the operations specified by the user.

Input File Format:

Your input file consists of three columns:

American Name: US- based Pokemon name
Pokedex number: ID number, which is unique for each pokemon
Japanese Name: Japanese name of Pokemon

For example:

Abra 63 Casey
Aerodactyl 142 Ptera
……

Your program should be capable of performing the following operations on BST:

1. Adding to a BST
2. Searching a BST based on ID (Pokedex number)
3. Visit a BST in pre, in and post order
4. Remove from the BST (In Phase 2)
5. Copying a BST (In Phase 2)

Start implementation by adding to the BST and then gradually move to number 2,3,4 and 5

Requirements

Your program should take one input.txt file in the format above as command line argument. Once you run the program, your terminal should print the menu listed several operations:

1. Search - Ask user to enter pokedex number. Based on the number you have to print the US name and Japanese name
2. Add
3. Copy
4. Remove
5. Print
6. Quit

When you run the program, your program should first read the file and add the pokémon to the BST in the correct order where for each node, its left child is smaller and right child is larger than the value stored in that node.

Implementation Details:

- Your BST class will be templated with two types:
1. ItemType - return type of your search(Pokemon object)
2. Key Type - The type you can search on(Pokedex number)

Which means when you search for ID 25, you should return the entire entry corresponding to that Pokedex number i.e. the US Name, ID nad Japanese name for the Pokemon with that Pokedex Number.

Refer to the BST interface, which has been given in wiki page.

- You have to implement the overloaded comparison operators for your Pokemon

class For example:

```
bool poke::operator==(int id)
{
   if(m_id==id) return(true);
   else return(false);
}

bool poke::operator>(int id)
{
   if(m_id>id) return(true);
   else return(false);
}

bool poke::operator==(const poke& rhs)
{
   if(m_id==rhs.m_id) return(true);
   else return(false);
}

bool poke::operator>(const poke& rhs)
{
   if(m_id>rhs.m_id) return(true);
   else return(false);
```

- You have to make a copy of the BST using copy constructor. Once you make a copy, you have to ask user whether they want to use copy BST or normal BST. When you use copy BST, your original BST should remain unchanged.

Binary Node Class

Node.h
Node.cpp

Pokemon Class

- getters and setters for US name, Japaneese name and Id,
- Overloading operators
- Helper functions

BST Class

- all the methods listed on BST interface
- You also required to add the private functions

Executive Class

- reading from the file
- printing menu
- call the appropriate function according to user input.