Homework #2: Testing the performance of binary search trees.

****

Testing the performance of binary search trees for managing a large number of dates:

Step #1. Implement a *Tree* class for managing dates:

Your task in Step #1 is to revise the ***Tree class*** in the simple Tree C++ project regarding the implementation of a simple binary search tree class that can store integers to implement a new ***Tree class*** that can store date information. You should create a revised project and craft the source code files in the project as described in the following:

- **DateType.h**:
  Use the same **DateType.h** from your programming #2, which specifies the logical interface of `DateType` class. Incorporate it into the project.
- **DateType.cpp:**
  Use the same **DateType.cpp** from your programming #2 that implements the member functions of the `DateType class`. Incorporate it into the project.
- **Tree.h:**
  Revise things related to the declaration of *Tree class* as described in the following so that it can manage dates instead of integers. (i) Modify the *TreeNode* structure such that the *Value* field of a Tree node stores a *DateType* object, instead of an integer. (ii) Change the type of the *Val* parameter in all the related member functions in the *Tree* class from *int* into *DateType* too.
- **Tree.cpp:**
  Revise things related to the implementation of *Tree class* as described in the following so that it can manage dates instead of integers. (i) Change the type of the *Val* parameter in the implementation of all the related member functions in the *Tree* class from *int* into *DateType*. (ii) Similarly change the type of the related local variables in the implementation of these member functions from *int* into *DateType* too.

- **main.cpp:**
  Revise the main function in according to Step #2 below to implement options for testing the performance of binary search trees.

Step 2. Implement the testing options:

In your main function in main.cpp, you should **first** declare a local *tree* object *T*, and ***then*** implement a loop that repeated do the following in each iteration : (i) displays two options X and Y, (ii) ask the user to pick one of the options, and (iii) do the following things according to the option selected by the user:

- Option X (do random ***Insert*** for *n* times):

When the user selects this option, your program should ***(i)*** call the *Clear* method to empty the binary search tree *T*, ***(ii)*** ask the user to enter a natural number *n*, ***(iii)*** declare a local *DateType* object *d*, and ***(iv)*** set up a loop to go through *n* iterations and in each iteration call *d.SetRandomDate( )* to set a random date and then call the ***Insert(d)*** method to insert the date in *d* into the binary search tree *T*.

- Option Y (do random ***Delete*** for *n* times):
  When the user selects this option, your program should ***(i)*** ask the user to enter a natural number *m*, ***(ii)*** declare a local *DateType* object *d*, and ***(iii)*** set up a loop to go through *m* iterations and in each iteration first call *d.SetRandomDate( )* to set a random date and then call ***Delete (d)*** to try to remove the date in *d* from the binary search tree *T*.

Step 3. Experiments:

A. **Test and report the time needed for *n* insertions into a binary search tree**: Try option X several times and use different values of *n* from 1000, 10000, 100000, and up to at least 10,000,000 or higher. Each time use your watch to roughly estimate the amount of time option X takes (to insert *n* random dates into a **binary search tree**). **Record and report your findings.**

B. **Right after Experiment A, test and report the time needed for *m* deletions in binary search tree of about *n* nodes (where *n* is the value you used for Option X in the very end of Experiment A)**: Try option Y several times now using different values of *m* from 1000, 10000, 100000, and up as you did in Experiment A above. Each time use your watch to roughly estimate the amount of time option Y takes (to remove *m* random dates from the **binary search tree** established by Option X in the very end of Experiment A). **Record and report your findings.**

Step 4. Reflection and analysis:

A. **About the time needed for *n* insertions into an initially empty binary search tree**: What do you think is the relationship between the size *n* and the amount of time needed? Why? **Record your thoughts/analysis.**

B. **About the time needed for *m* deletions in a binary search tree of about *n* nodes**: What do you think is the relationship between the size *n* and the size *m* and the amount of time needed? Why? **Record and report your findings.**

Submit your work

- Record all your experimental findings in Step 3 and your thoughts in Step 4 above in a WORD document. Submit the WORD document under Canvas.
- Compress your entire Program folder into a zip file and upload it through Biola Canvas.

- Carefully fill out this self-evaluation report and upload it through Biola Canvas. Note that you will receive no point for missing the self-evaluation report or missing the integrity review in the report.