CS 201 Data Structures Library Phase 2 Due 10/15

Phase 2 of the CS201 programming project, we will be built around a balanced binary search tree. In particular, you should implement treaps using the insertion and deletion algorithms discussed in class.

The public methods of your class should include the following methods, where keytype indicates the type from the template. The treap is ordered on the keys (binary search tree order) and on the priorities (heap order). The keys will be unique. In the case of equal priorities, place the smaller key as the parent.

Function	Description	Runtime
Treap();	Default Constructor. The tree should be empty	O(1)
Treap(keytype k[], float p[], int s);	For this constructor the tree should be built using the arrays k and p containing s items of keytype and s floats in the range [0 … 1].	O(s lg s) expected
~Treap();	Destructor for the class.	O(n)
float search(keytype k);	Traditional search. Should return the priority associated with the key. If the key is not stored in the tree then the function should return -1	O(lg n) expected
void insert(keytype k, float p);	Inserts the node with key k and priority p into the treap.	O(lg n) expected
void insert(keytype k);	Inserts the node with key k and generates a random priority in the range [0…1] to use to insert the key.	O(lg n) expected
int remove(keytype k);	Removes the node with key k and returns 1. If the node containing key k has more than one child, replace the key k and priority by the key and priority of the predecessor. The If key k is not found then remove should return 0.	O(lg n) expected
int rank(keytype k);	Returns the rank of the key k in the tree. Returns 0 if the key k is not found. The smallest item in the tree is rank 1.	O(lg n) expected
keytype select(int pos);	Order Statistics. Returns the key of the node at position pos in the tree. Calling with pos = 1 should return the smallest key in the tree, pos = n should return the largest.	O(lg n) expected
keytype successor(keytype k)	Returns the key after k in the tree. If the key has no successor, return k.	O(lg n) expected
keytype predecessor(keytype k)	Returns the key before k in the tree. If the key has no predecessor, return k.	O(lg n) expected
int size();	returns the number of nodes in the tree.	O(1)
void preorder();	Prints the keys of the tree in a preorder traversal.	O(n)
void inorder();	Prints the keys of the tree in an inorder traversal.	O(n)
void postorder();	Prints the keys of the tree in a postorder traversal.	O(n)

Your class should include proper memory management, including a destructor, a copy constructor, and a copy assignment operator.

For submission, all the class code should be in a file named Treap.cpp. Create a makefile for the project that compiles the file Phase2Main.cpp and creates an executable named Phase2. A sample makefile is available on Blackboard. Place both Treap.cpp and makefile into a zip file and upload the file to Blackboard.

☐ Create your Treap class
☐ Modify the makefile to work for your code (changing compiler flags is all that is necessary)
☐ Test your Treap class with the sample main provided on the cs-intro server
☐ Make sure your executable is named Phase2
☐ Develop additional test cases with different types, and larger trees
☐ Create the zip file with Treap.cpp and makefile
☐ Upload your zip file to Blackboard

No late submissions will be accepted. There will be an opportunity to resubmit by 10/30. Resubmissions will have a 20 point penalty.