CS 201 Data Structures Library Phase 3 Due 4/18

Phase 3 of the CS201 programming project, we will be built around heaps. In particular, you should implement both a standard binary heap and binomial heap. You will implement a class for each heap type.

You should create a class named Heap for the binary heap with public methods including the following (keytype is the type from the template). You should use your dynamic array class for the heap storage.

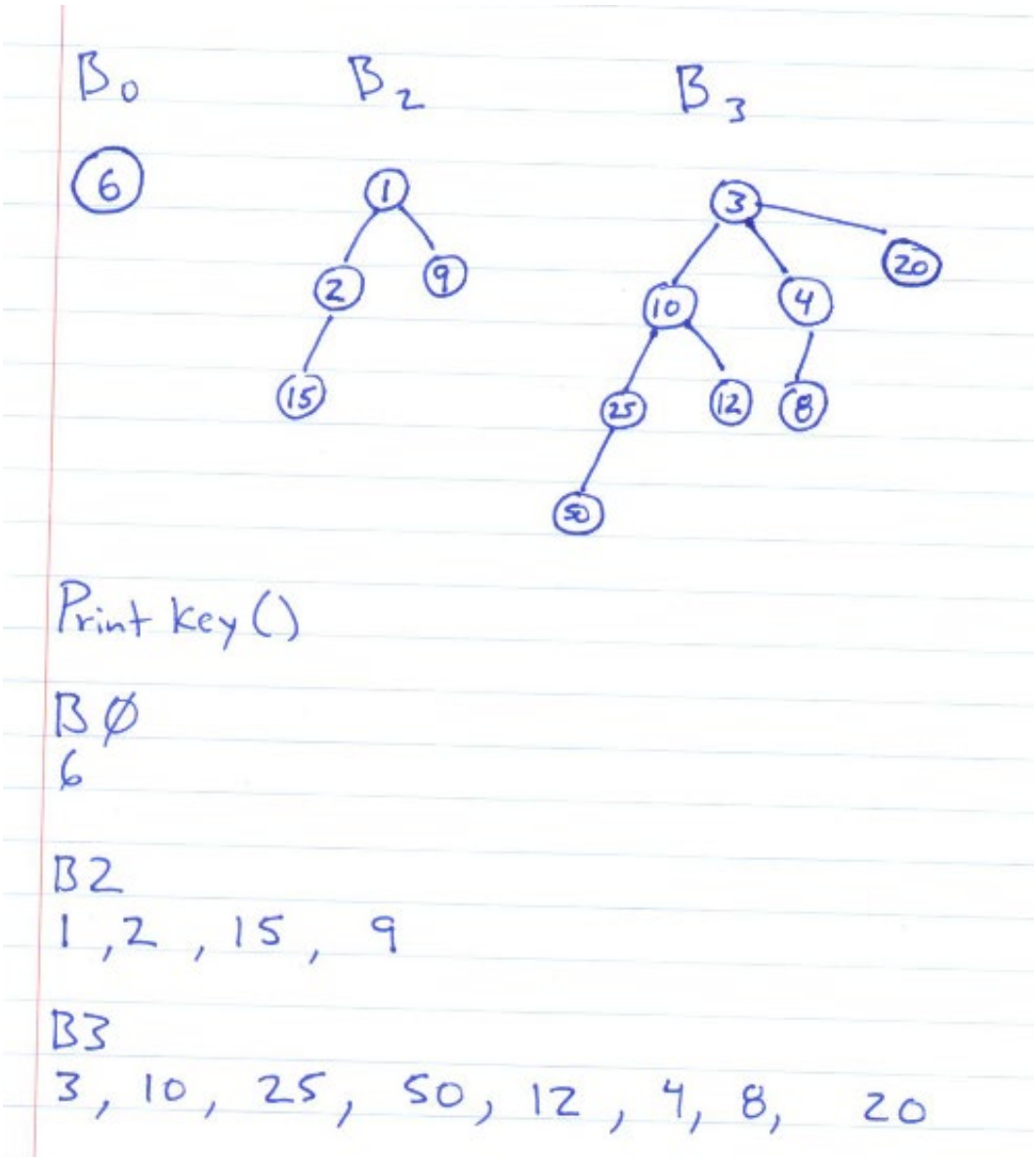
Function	Description	Runtime
Heap();	Default Constructor. The Heap should be empty	O(1)
Heap(keytype k[], int s);	For this constructor the heap should be built using the array K containing s items of keytype. The heap should be constructed using the O(n) bottom up heap building method.	O(s)
~Heap();	Destructor for the class.	O(1)
keytype peekKey();	Returns the minimum key in the heap without modifiying the heap.	O(1)
keytype extractMin();	Removes the minimum key in the heap and returns the key.	O(lg n)
void insert(keytype k);	Inserts the key k into the heap.	O(lg n)
void printKey()	Writes the keys stored in the array, starting at the root.	O(n)

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

You should create a class named BHeap for the binomial heap with public methods including the following (keytype is the type from the template). You should use dynamic allocation for the binomial trees. Note that in order to perform peekKey in O(1) time, you will need to maintain a pointer to the minimum value in the heap.

Function	Description	Runtime
BHeap();	Default Constructor. The Heap should be empty	O(1)
BHeap(keytype k[], int s);	For this constructor the heap should be built using the array K containing s items of keytype. The heap should be constructed using repeated insertion.	O(s)
~BHeap();	Destructor for the class.	O(n)
keytype peekKey();	Returns the minimum key in the heap without modifiying the heap.	O(1)
keytype extractMin();	Removes the minimum key in the heap and returns the key.	O(lg n)
void insert(keytype k);	Inserts the key k into the heap.	O(lg n)
void merge(BHeap<keytype> &H2);	Merges the heap H2 into the current heap. Consumes H2.	O(lg n)

void printKey()	Writes the keys stored in the heap, printing the smallest binomial tree first. When printing a binomial tree, print the size of tree first and then use a modified preorder traversal of the tree. See the example below.	O(n)



For submission, all the class code should be in a files named Heap.cpp and BHeap.cpp. Create a makefile for the project that compiles the file Phase3Main.cpp and creates an executable named Phase3. A sample makefile is available on Blackboard. Place Heap.cpp, BHeap.cpp and makefile into a zip file and upload the file to Blackboard.

☐ Create your Heap and BHeap classes
☐ Modify the makefile to work for your code (changing compiler flags is all that is necessary)

- ☐ Test your Heap and BHeap classes with the sample main provided on the cs-intro server
- ☐ Make sure your executable is named Phase3
- ☐ Develop additional test cases with different types, and larger arrays
- ☐ Create the zip file with Heap.cpp, BHeap.cpp and makefile
- ☐ Upload your zip file to Blackboard

No late submissions will be accepted.