CSCI-311 **Assignment 10**

Download Lab10.tar from Blackboard.

To untar use this command: tar -xvf Lab10.tar

Your task is to implement a priority queue using a **min**-heap. The smallest-priority item will be the root of the heap.

You are provided the following files: *item.h, priorityqueue.h, priorityqueue.cpp, main.cpp* and Makefile.

Class ***PriorityQueue*** has the following data members and constructors:

PriorityQueue() //default constructor: creates arrays pointed to by *aheap* and *keys* of **capacity 2**.

PriorityQueue(Item *array, int length) //takes a pointer to an input array with Item objects and length of array

// and builds a heap from it using O(n)-time approach

~PriorityQueue(); //destructor of the class: deallocates space taken by the heap

private:

Item* ***aheap*** = NULL; //a pointer to an array that will be created in the heap of Item objects

int *<b><i>keys</i></b> = NULL;//a pointer to an array whose indices represent keys of Items in the heap,

//and keys[k] returns index j such that Item at *aheap*[j] has key equal to k.

int ***size***; //total of items in the heap (smaller or equal to the length of the array)

int ***capacity***; // the actual size (length) of the array, to which *aheap* points

int ***totalKeys***;// the maximum number of keys that has been used in the heap

int ***capacityKeys***; // the actual size (length) of the array pointed by *keys*

You need to implement the following public member functions of PriorityQueue class:

Member function	**Description**	Tests
int getCapacity()	Returns *capacity* of *aheap*	t01
int getSize()	Returns *size*, to which *aheap* points	t01
void print()	Prints three lines: 1) priorities of items at indices 0, 1, 2, …, size – 1 of *aheap* 2) keys of items at indices 0, 1, 2, …, size – 1 of *aheap* 3) indices stored in the array pointed by *keys;* print all keys starting with index 0 and ending with index ***totalKeys - 1*** Format: value followed by space, and print ***endl*** after each line You will need this function for debugging	all

void reheapifyUp(int i)	This is a recursive function. Given an index of the array, it places the Item at that index into the correct position within the heap by recursively swapping with a parent if necessary	t06,t07
void reheapifyDown(int i)	This is a recursive function that given an index of the array, it places the Item at index i into the correct position within the heap by recursively swapping with the smallest of the two children (if children are equal, then swap with the left child)	t08,t09
void pop()	Removes min-value from the heap, if the heap is empty, does not do anything. Set *keys* of the removed item to -1.	t08,t09
Item getMin()	Returns the min item in the heap (but does not remove it from the heap)	
bool push(int akey, int apr)	Given an object of class Item, it adds this item to the heap. It needs to check the capacity of the heap: if the size of the heap is equal to it's capacity, then this function must allocate new array (increase capacity by 2, i.e. new capacity is twice as big as the old capacity), copy to this array the content from *aheap*, and then deallocate space taken by the old *aheap* and only then add a new item to *aheap*. Do same for *keys.* Increment *size* and *totalKeys.* Need to check: *akey* must be equal to the current (old) *totalKeys*. If it is not, return false (meaning the item was not enqueued into the heap).	t02-t05
bool updatePriority(int akey, int apr)	Uses *akey* to find Item with this key and decrease this item's priority to the new value equal to *apr.* Need to check if an item with *akey* is still in the heap (keys[akey] is not -1) before actually updating priority. Need to check: if *apr* is greater than the old priority, return *false* and do nothing. We only can decrease priorities in the Min-heap.	t06, t07

Test ***t10*** checks all member functions.

Grading: Functions reheapifyUp/Down are worth 25pts each; functions print, push, pop and updatePriority are worth 5pts each. Test t10 is worth 30pts.

Submission:

Submit *item.h, priorityqueue.h, priorityqueue.cpp* to turnin.