CISS350: Data Structures and Advanced Algorithms
Assignment 14

Objectives:

1. Implement a max heap.
2. Implement a priority queue.

Q1. [Max heap]

Implement the max heap functions (see below) where the heap is implemented using vectors of integers. (Vectors as in `std::vector`.) Write the max heap functions in `intmaxheap.h` and `intmaxheap.cpp`.

Refer to the notes (and textbook) for details on max heap operations.

The following describes all the max heap functions that must be implemented. Also, implement `operator<<` so that you can print your vector while testing your code (see below).

```
int x;
std::vector< int > heap;

maxheap_insert(heap, 5);        // [5]
maxheap_insert(heap, 7);        // [7, 5]
maxheap_insert(heap, 9);        // [9, 5, 7]
std::cout << heap << std::endl; //  prints "[9, 5, 7]"

int a = maxheap_delete(heap);   // [7, 5]
                                // a = 9
x = maxheap_max(heap);          // x = 7. Root is not deleted.

heap[0] = 1;                    // [1, 5]
maxheap_heapify_down(heap, 0);  // [5, 1]

heap[1] = 10;                   // [5, 10]
maxheap_heapify_up(heap, 1);    // [10, 5]

heap.resize(5);
heap[0] = 5;
heap[1] = 7;
heap[2] = 8;
heap[3] = 10;
heap[4] = 2;
maxheap_build(heap);            // [10, 7, 8, 5, 2]

heap.resize(5);
heap[0] = 2;
heap[1] = 6;
heap[2] = 8;
heap[3] = 10;
heap[4] = 5;
```

```
maxheap_heapsort(heap)          // [2, 5, 6, 8, 10]
std::cout << heap << std::endl; // prints "[2, 5, 6, 8, 10]"
```

Put the above in `main.cpp`:

```
#include <iostream>
#include "intmaxheap.h"

int main()
{
    // test code
    return 0;
}
```

(As always skeleton code needs to be modified/edited/corrected/etc.)

Q2. [Max heap]

Re-implement the max heap functions from Q1 to function templates for max heap. The name of the header file must be `maxheap.h`.

```
// File: maxheap.h
// This file contains the function templates for the max heap.

template < typename T >
void maxheap_insert(std::vector< T > & h, const T & x)
{
    // TO BE COMPLETED
}

// Etc.
```

To test the max heap function templates, first here a class for values to be places in the max heap:

```
// File: maxheap_value.h
class maxheap_value
{
public:
    // TO BE IMPLEMENTED

private:
    int priority;
    std::string s;
};
```

```
// File: maxheap_value.cpp
std::ostream & operator<<(std::ostream & cout, const maxheap_value & x)
{
    cout << '<' << x.get_priority() << ", " << x.get_s() << '>';
    return cout
}
```

Convert the test code from Q1 appropriately. (The string part of the max heap values are completely arbitrary.)

```
maxheap_value x;
```

```
std::vector< maxheap_value > heap;

maxheap_insert(heap, maxheap_value(5, "a")); // [<5, "a">]
maxheap_insert(heap, maxheap_value(7, "b")); // [<7, "b">, <5, "a">]
maxheap_insert(heap, maxheap_value(9, "c")); // [<9, "c">, <5, "a">, <7, "b">]

max_heap_value a = maxheap_delete(heap); // [<7, "b">, <5, "a">]
                                         // a = <9, "c">
x = maxheap_max(heap);                   // x = <9, "c">. Root is not deleted.

heap[0] = maxheap_value(1, "d");         // [<1, "d">, <5, "a">]
maxheap_heapify_down(heap, 0);           // [<5, "a">, <1, "d">]

heap[1] = maxheap_value(10, "e");        // [<5, "a">, <10, "e">]
maxheap_heapify_up(heap, 1);             // [<10, "e">, <5, "a">]

// ...
```

Put the above test code in `main.cpp`:

```
#include <iostream>
#include "maxheap_value.h"
#include "maxheap.h"

int main()
{
    // test code
    return 0;
}
```

(As always skeleton code needs to be modified/edited/corrected/etc.)

Q3. [Priority queue]

Implement a class template for maximum priority queues. Objects in the maximum priority queue must have a `get_priority()` method that returns an integer value.

`MaxPriorityQueue< T > q`	`constructor`
`q.insert(value)`	`insert value into q`
`q.delete()`	`remove and return the value at the front of q`
`q.max()`	`return reference to value at the front of q`
`q.clear()`	`remove all values from q`
`q.size()`	`number of values in q`
`q.is_empty()`	`true iff q has size 0`
`q0 = q1`	`the obvious assignment operator`
`q0 == q1`	`the obvious comparison operator`
`MaxPriorityQueue< T > q0(q1)`	`the obvious copy constructor`

Note that you should use Q2.