Comp 121: Data Structures (Spring 2002) **Programming Assignment 1 (** Due: 2002/02/21)

Objective: To gain experience in implementing abstract data types (ADT's).
Goal. To implement the *priority queue* ADT as an object template in C++. To use this ADT as the basis of a sorting program.

The priority queue (*pqueue*) abstract data type is **specified** as follows:

AbstractDataType pqueue{
Instances: finite collections of zero or more elements
Operations:

Create( ):	Create an empty priority queue
Destroy( ):	Erase a priority queue
IsEmpty( ):	Return *true* if the priority queue is empty; *false* otherwise
Min( ):	Return the minimum element in the priority queue
DeleteMin( ):	Delete the minimum element in the priority queue, and return this element
Insert(x):	Insert x into the priority queue

}

You are to **implement** this ADT in the C++ programming language. I.e., you are to define a priority-queue class template

```
template pqueue <class T>{
        .
        .
};
```

such that declarations (known as **instantiations** of the class template) of the form

```
pqueue <int> PQ1;
pqueue <myClass> PQ2;
```

would create an instance PQ1 of a priority queue capable of holding integers, and an instance PQ2 of a priority queue that is capable of holding myClass objects (where "myClass" is an object that you have previously defined).

Algorithms/ data structures. Implement your priority queue as a *linked list.* You may either choose to keep this list sorted at all times (in which case the minimum element is easily found), or you may keep it as an unsorted list and search for the minimum element as needed. Jutify your design decision.

Priority-queue sort. Priority queues can form the basis of a *sorting* algorithm – given a file of data-items that are to be sorted, we can insert these elements into a priority queue, and then repeatedly delete the minimum element from the priority queue until it is empty. Test your priority queue implementation by using it to sort a number of *points,* where a point is specified according to its x- and y- coordinates, and points are compared according to their distance from the origin (the point with coordinates *(0,0)*). Assume that you are given the points to be sorted in a file, which contains the number of points in the file followed by a pair of coordinates for each point.

Program structure. Both the declaration of the pqueue template, and the implementation, need to be in the same file (call this file pqueueTemp.h). You should also define a class *point* for representing points – this class should be declared in file point.h, and implemented in point.cpp. Your "main" program should be the one that implements the priority-queue based sorting routine, and should be in a separate file. With all programming assignments, *you must submit a cover sheet, and a test-plan, design plan, etc.*