Assignment 3a. **Heaps and recurrence relations, second chance.**

Given March 4, due March 26 (Monday).

1. A queue has operations \texttt{enqueue(item x)}, which adds the object of data type \texttt{item} to the end of the queue, and \texttt{item dequeue()}, which returns the item at the head of the queue and removes this item from the queue. There are two common ways to implement a queue. One uses an array (a “circular array”) and the other uses a linked list. We want to simulate the mob scene at the airport checkin line. Customers arrive with a departure time. Normally, people wait in the line (queue), but anyone whose flight leaves in less than 30 minutes gets the next available server. For the simulation, we need a function that returns a customer leaving in less than 30 minutes, if there are any, and removes that customer from the queue. Explain how to implement this by threading together a heap and a queue. All the queue operations should take $O(1)$ time while the heap operations may take $O(\log(n))$ time. Note that when a customer who is not first in line gets the next server because of the thirty minute rule, you have to remove an item from the queue that is not the head. This should determine whether to use an array or linked list for the queue. There is a quick review of queues in CLR somewhere.

2. A server has many jobs pending, each with a completion time and a category. There are \textit{k} categories, which may be numbered 0, ..., \textit{k} – 1. We need a data structure and collection of methods for the following operations

   \begin{verbatim}
   insert( job x) // put the job x into the system
   job deleteMax() // return the job with the largest completion time
   job deleteMax(category j) // return the job from category j with the
   // largest completion time
   \end{verbatim}

   (Note, in C++ and Java it is possible to have two procedures with the same name if they have different calling arguments. The “signature” of a procedure is the name of the procedure and the types (in order) of its calling arguments.) The data type \texttt{job} has fields

   \begin{verbatim}
   job.time // the completion time
   job.cat // the category, an integer from [0,1,...,k-1]
   job.data // the rest of the information about the job.
   \end{verbatim}

   Use \textit{k}+1 heaps, one for each category and one more for the overall \texttt{deleteMax} operation. Be careful to check, when you insert a job into the category \textit{j} heap, whether is is the new maximum in category \textit{j}. Also, when you delete the max element from category
\[ j, \text{ you must also delete it from the overall max heap (we've done that before). For efficiency, the overall max heap should only have the } k \text{ maxima from the the category heaps, not all the pending jobs.} \]

3. The Strassen algorithm for matrix multiplication leads to the recurrence relation

\[ W(n) = 7 \cdot W\left(\lfloor \frac{n}{2} \rfloor \right) + 21 \cdot n^2, \quad W(1) = 1. \]

a. Write \( J(k) = W(2^k) \). Write a recurrence relation for \( J(k) \) in terms of \( J(k - 1) \), unroll it, and compute the resulting geometric sum to find an explicit formula for \( J(k) \).

b. Argue, possibly using induction, that \( W(m) \leq W(n) \) if \( m \leq n \).

c. Combine a. and b. to show that \( W(n) = \Theta(n^p) \) where \( p = \log(7) \). This requires you to prove both an upper bound and a lower bound.

d. The standard algorithm for matrix multiplication requires exactly \( n^3 \) floating point multiplications or additions. Since \( p < 3 \) and \( W(n) \) is the number of floating point multiplications and additions needed by the the Strassen algorithm, Strassen will be faster for large enough \( n \). Use your answer to a. to estimate the first value of \( n \) for which the Strassen algorithm is faster.