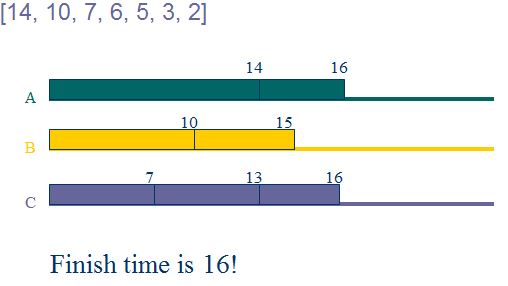
**COP 3530, Fall 2015**  
Data Structures and Algorithms  
Assignment 3: Longest Processing Time  
**Due: November 9, 2015, 11:59PM**

**Description:**  
 Scheduling is an NP-hard optimization problem. However, it is applicable in a vast number of fields and situations. An often used heuristic for this problem is the Longest Processing Time algorithm. You are required to implement the Longest Processing Time algorithm as follows. Your algorithm will be implemented in two versions using two different representations of a minimum priority queue. The first implementation will use an array-based min-heap. The second implementation will use a pointer-based min height-biased leftist tree.



For both implementations of your Min Priority Queue data structure, you must implement a constructor, destructor, empty, size, push, top, and pop functions. You may also implement any helper function(s) at your own discretion.

* **The meld function of your leftist heap implementation must be recursive**.
* The empty function returns a bool of whether or not the priority queue is empty.
* The size function returns the current number of elements in the priority queue.
* The push function inserts an element into the priority queue.
* The top function returns the element with the min priority.
* The pop function removes the element with the min priority.

You will run the dataset **on each version** and output the finishing time.  
The program will also print the schedule.

**Sample input and output (User Input is represented in RED font ):**

Enter number of job(s):  
7  
Enter number of machine(s):  
3  
Enter Processing Time(s):  
14  
7  
10  
6  
2  
3  
5  
  
Min Heap Finishing Time: 16  
Schedule:  
Machine 1: 14, 2

Machine 2: 10, 5

Machine 3: 7, 6, 3  
Time Elapsed: 3.882e-06

Height Biased Leftist Tree Finishing Time: 16

Schedule:

Machine 1: 14, 2

Machine 2: 10, 5

Machine 3: 7, 6, 3  
Time Elapsed: 3.927e-06

Note: Your elapsed times will likely differ from these.

You are not allowed to use any heap or leftist tree facilities in the C++ STL.

We will test your submissions by following these steps (commands on thunder.cise.ufl.edu):

1. tar xvf “<LastName\_FirstName>\_UFID.tar”

2. make

3. ./LPT < our\_input

**Deliverables:**

* Your submission tarball(.tar archive file) should be named <LastName\_FirstName>\_UFID.tar and must contain the following files: LPT.cpp, Makefile, and <LastName\_FirstName>\_UFID\_report.pdf
* A PDF document containing your own test cases, test results, and any special diagnostics you utilized. This file shall be named <LastName\_FirstName>\_UFID\_Report.pdf

**PLEASE NOTE** that **ALL** submissions **MUST** compile on **thunder.cise.ufl.edu** by using your Makefile. It is highly recommended that you upload your source code to your CISE account and test it on the thunder. Secure Remote Access to CISE machines is available. Please visit this link for more information, <https://www.cise.ufl.edu/help/access/remote>. We will not debug your source code or makefile. Please refer to and make note of the submission rules and policies before submitting, <http://cise.ufl.edu/class/cop3530fa15/SubmissionRules.htm>. After submitting on Canvas, you should verify that your submission was successful by downloading it from Canvas (to a separate location), and successfully un-tarring, compiling, and running it. YOU ARE NOT DONE UNTIL YOU DO THIS.