Due: Monday, March 16th at 11PM

Description
Create a pseudo-kernel to handle memory management. You will use linked lists to hold the pages for each task. You will have twelve lists, two for each of six tasks. One list holds all of the pages assigned to the task, the second one holds the pages loaded into local memory. Each task can have no more than three pages loaded into local memory.

You will test the operation of your kernel. At the beginning of the run, you will determine how long each of the six tasks will execute. Use a timing routine to measure the execution of each task. Each task will start with at least four pages of memory, three of which will be loaded into local memory. At random intervals during its execution, the task will either add a new page to its memory or access a page that is not loaded (requiring replacement). When a task adds a page, it is added to both lists (meaning the new page is accessed into local storage), so an existing page must be replaced. The replacement is handled by using the list_replace( ) function on the linked list holding the pages loaded in local memory. You will choose which page to replace using the Least Recently Used policy. The Least Recently Used policy will be implemented in a separate routine; that routine will call the list_replace( ) function.

At least four times during your sample run, your kernel should list the pages on each linked list. Identify the lists by task number and which list (total pages or pages in local storage). At the start of the run, list the initial number of pages of each task and how long each task will execute.

When the task’s time is over, delete all of the elements from the linked lists before clearing them (we want to prevent storage orphans!).

Turn in
1. Your entire code, including the kernel and any included files that you wrote.
2. The output from a sample run.

Turn in your assignment to csi500@cs.albany.edu.