CS 3413

Assignment 6

Due Date: November 4th, 2019 at 9:30 am

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

Your solution should be submitted via D2L. All solutions are to be written in C.

Memory is allocated for a given process when a user starts a program. The OS can give each process a different amount of memory based on the amount requested when the process is created. So, requests to the OS are received from applications to allocate (creation) or to free (termination) memory. The OS services the request and gives the new process the memory requested according to 3 popular algorithms:

- First fit: Satisfy the request from the first available free memory block that is large enough to accommodate the request.
- Best fit: Satisfy the request from the free memory block that is large enough to service the request and small enough that it has the smallest fragmented block.
- Worst fit: Satisfy the request from the free memory block that is large enough to service the request and creates the largest fragmented block.

Consider the following input format:

N*1 500*
T*7*
S

N indicates the process (*1*) that is being created needs memory (500 bytes). The result of this operation is either successful (indicated by a memory address returned that references the new memory) or a failure (indicated by a NULL memory address returned). The **T** operation is terminating the process (process 7 in this case) and freeing the memory that was assigned to the process. The **S** operation is to stop the program and print the report.

You are to write a program (no pthreads! – YAY) that will simulate the memory allocations/frees of the operating system. Your program will read input from `stdin` and produce output on `stdout`. Each line will be one of the three forms above.

Your program will take the total amount of memory in your system as a command line parameter "-s #". You have to implement all 3 algorithms with a command line option "-f", "-b", or "-w" to select which algorithm is used.

Output generated is the following (either message for each allocation line of input):

- If an allocation fails:
  - **Process # failed to allocate *x* memory**
- If a free fails:
  - **Process # failed to free memory**

Regardless of success or failure, your program will continue to service ALL requests on stdin. If a process fails to be allocated, then its corresponding terminate request is ignored.

At the end, you are to print a short report of the overall results:

  - **Total Processes created #, Total allocated memory #, Total Processes terminated #, Total freed memory #, Final memory available #, Final smallest and largest fragmented memory sizes**

Where each # are the sums for the given operations.

Warning! Be careful … there are no limits on how many processes can be created/terminated or how much memory you may have when you start ☺