Project OS4

Due Date: November 19

Purpose

This simulation will let you implement the first-fit memory allocation scheme, and get a glimpse of its performance.

This is an **individual** project.

Problem

The latest computers from GousJunk Machines will use a new version of the company's operating system, osX_{10}*LinWin*. The designers have not yet decided on the memory allocation method beyond using a contiguous scheme. You, a recognized expert in the field, have been hired to run a simulation to test first-fit memory allocation for possible use in the new system.

Input

The program should first prompt the user for a file name. The file will contain any number of lines with the format:

<op> *<PID>* *<size>*

where *op* is either 'A' or 'D', for allocate or deallocate, respectively; *PID* is the process id (an integer); and *size* is the size of the job in bytes. If the *op* is 'D', the input line will not include the job's size. An *op* of 'Q' indicates the end of the file.

Assume that the input file is error-free, **except** that an input of 'D' may be in the file for a process that did not previously fit into memory. In this case, there is nothing to delete (i.e., ignore the 'D'.)

Output

After all the input has been read and processed, the program should display the following information:

- A table showing the final configuration of memory. The (aligned) table should show the PID of any process currently in memory as well as the beginning and ending addresses and size of its allocation. For example:

Memory Table

PID	Begin	End	Size
1	0	19999	20000
10	20000	69999	50000
12	70000	99999	30000
6	100000	131399	31400
Free	131400	231399	100000
8	231400	etc.	

- Number of processes (and their PIDs) that did not fit in memory
- Total amount of memory in use at the end of the simulation
- Total amount of memory remaining

- Number of contiguous spaces (memory blocks) remaining
- Average size of remaining memory blocks

Specifics

- Assume the size of main memory is 1M (2^{20}). This is unrealistic, but should be scalable to any size.[1] Processes can be of any size. The system has no compaction (garbage collection) capabilities, aside from coalescing contiguous blocks when memory is deallocated.
- The program should be designed using OOP principles. For example, you will need some sort of data structure (queue? vector?), which should be a separate class/ADT; alternatively, you can use the Standard Template Library (STL). The simulation portion should be defined in its own class containing relevant data members and methods.
- A sample input file will be available on the course Web page.

Notes

This is a much easier project than the previous three; 200 or so lines of code ought to do it.

Submit your source code by emailing it to me as usual. Submit hard copy of your source code in class on November 20^{th}.

Bad magic number
– Unix error message

[1] This means that, assuming you have a named constant for this, if you change the constant's value, your program should work for any size memory.