CSS-430 : Operating Systems : P4

**Contiguous Memory Allocation**

Section [9.2](https://jigsaw.vitalsource.com/books/9781119320913/epub/OPS/c09.xhtml#c09-sec-0002) of the OSC book described several algorithms for contiguous memory allocation: First-Fit, Best-Fit and Worst-Fit.

This project asks you to manage allocations within a memory pool of size MEMSIZE. Your program support five different requests:

**1.** **Allocate** N bytes for a process using one of the 3 allocation algorithms

**2.** **Free** all allocations for a given process

**3.** **Show** the status of the memory pool – allocated and free blocks

**4.** **Read** a script – a sequence of commands from a file, and execute them

**5.** **Compact** the allocations, making them into one contiguous block. (This somewhat resembles the operation of a mark-sweep garbage collector in C#)

MEMSIZE has a value of 80. So we can think of the memory pool as holding 80 bytes. (If you prefer, you can think of it as holding 80 KBytes, where allocations are made in multiples of 1 KByte)

Processes are named as a single letter, A thru Z.

Here is an example of the Show command after some allocations and frees:

AAAAAAAAAA..........BBBBBBBBBBFFFFFFFFFFFFFFFGGG..CCCCCH..............DDDDD.....

(MEMSIZE was chosen as 80 so the display would fit nicely onto our console screen)

The 5 command formats are as follows:

* **A <name> <size> <algo>**  
  Allocate <size> bytes for process <name> using algorithm <algo>. <algo> can be any of F for First-Fit, B for Best-Fit or W for Worst-Fit.  
  Eg: A P 20 F Allocate 20 bytes for process P using First-Fit  
  Eg: A X 14 B Allocate 14 bytes for process X using Best-Fit
* **F <name>**  
  Free all the allocations owned by <name>  
  Eg: F P Free all allocations owned by process P
* **S**  
  Show the state of the memory pool
* **R <file>**  
  Read the script in the file called <file> and execute each command.  
  Eg: R MEMO.TXT
* **C**  
  Compact the memory pool, *sliding* all allocations to lower addresses so they become one contiguous block, and so that all the free space lies to the right as one contiguous block

Here is an example script:

A A 10 F

A X 10 F

A B 10 F

A X 20 F

A C 5 F

A X 15 F

A D 5 F

F X

The first Show command produces:

AAAAAAAAAAXXXXXXXXXXBBBBBBBBBBXXXXXXXXXXXXXXXXXXXXCCCCCXXXXXXXXXXXXXXXDDDDD.....

After the “F X” command, it becomes:

AAAAAAAAAA..........BBBBBBBBBB....................CCCCC...............DDDDD.....

**Compaction**

Here is an example pool before compaction:

AAAAAAAAAA..........BBBBBBBBBBFFFFFFFFFFFFFFFGGG..CCCCCH..............DDDDD.....

And after compaction:

AAAAAAAAAABBBBBBBBBBFFFFFFFFFFFFFFFGGGCCCCCHDDDDD...............................

The OSC book suggests the “Compact” command. In reality, however, an Operating System cannot move memory allocations around, willy-nilly. What if that memory included pointers, such as a linked-list. What would go wrong?

But virtual machines, as used by Java and C#, *do* track pointers. They can, and do, compact memory.

**Grading**

Here is the script we will run to grade submissions:

A A 10 F

A X 10 F

A B 10 F

A X 20 F

A C 5 F

A X 15 F

A D 5 F

F X

S

A E 25 F

A F 15 F

A G 3 B

A H 1 W

S

C

S

**Rubric**

Run the submitted program over the test script above. Check the results at 3 points in the run – where the S (Show) command occurs. Grade as follows:

* Correct results at first “Show” command : 15 points
* Correct results at second “Show” command : additional 5 points
* Correct results at third “Show” command : additional5 points