CS 218 DATA STRUCTURES
ASSIGNMENT 1
SECTION C and F
Fall 2022

DUE: Thursday, Sept 22, 2022

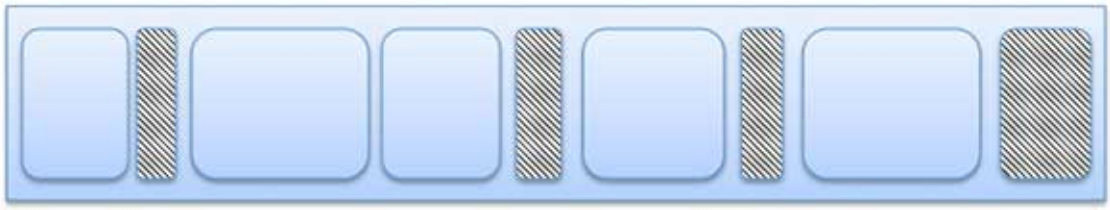
NOTE: Late submissions will not be accepted

TO SUBMIT: Documented and well-written structured code in C++ in the classroom. Undocumented code will be assigned a zero.

PROBLEM

In a computer system, the main memory (RAM) is a shared resource that all running programs share. The operating system serves as a resource manager. The main job of the operating system for memory management includes allocating free memory to the programs whenever they request it, reclaiming the memory of the programs after finishing their execution, and keeping a record of free and allocated memory.

Programs can claim memory at the start of the program and during its execution. So the memory reserved by one program may not be contiguous. Similarly, due to the dynamic nature of the programs, new programs are added to the system, and existing programs are removed after finishing their execution. This may cause memory fragmentation. **As processes are loaded and removed from memory, the free memory space is broken into little pieces called fragmentation.** In the figure below, shaded areas are free memory scattered in the main memory.



The job of an operating system is to allocate available byte(s) when a new process(program) is created and deallocate reserved memory when an existing program is finished. The Memory management module of the operating system maintains a **pool** of available bytes (or contiguous chunks of bytes) where new data of an existing program or a new program can be stored. It also stores the list of programs currently in execution and the parts of memory allocated to them. Whenever a new program is started, the memory manager uses available bytes of RAM to store the data of that program on the disk. Similarly, when a program is finished, the memory(bytes) allocated to that program are moved back to the Pool of available memory.

Whenever a program needs memory, the memory manager asks the total number of bytes required to store the data. The memory manager finds a chunk of contiguous memory in RAM and allocates that memory to the program. If the program asking for memory is a new program, a new program is added to the list of current programs. Otherwise, the information of the block of bytes allocated to the existing program is kept with the program information. A block maintains the Id of starting byte and the total number of

bytes. So a program in a memory management system has a collection of blocks where each block maintains the address of the next block of the program and the program stores the address of the first block. In order to make the whole process efficient, the memory management system maintains the *Pool of available blocks instead of available bytes*.

In this assignment, you will simulate the memory management system. The Memory management system includes ***RAM, a list of available blocks (Pool), and a list of Programs currently in execution***. We define the RAM as a one-dimensional array of ***bytes*** of size ***numOfBytes***. A ***pool*** of available blocks is kept in a linked list of ***block.*** A block has two fields: start_byte_ID and total_bytes. **Programs** are kept in a linked list ***Progs*** where each node has a program as data***.*** Each **program** has an ID, and a linked list of ***blocks allocated to that program.*** Asking memory requires claiming a sufficient number of bytes from the Pool, if available, and updating the Pool accordingly. Removal of a program requires removing the list of blocks allocated to the program and transferring the bytes assigned to this program back to the Pool.

Suppose we have a RAM of size 100KB. This means we will have byte ID's from 0 to 102400-1.

Figure 1 shows a link list of blocks called the Pool. The first available block is of 30 bytes starting from byte ID 420 and the last block is of 9900 bytes starting at byte ID 4000. Figure 2 shows a sample link list of Programs. There are 3 programs in the memory. The Id of program1 is P1 currently has acquired 260 bytes. It also tells that the data of the program is stored in 3 blocks. The first block starts at byte ID 0 and has 40 bytes, the second block has 20 bytes that start from byte ID 70 and the last block also has 200 bytes starting from byte ID 120.

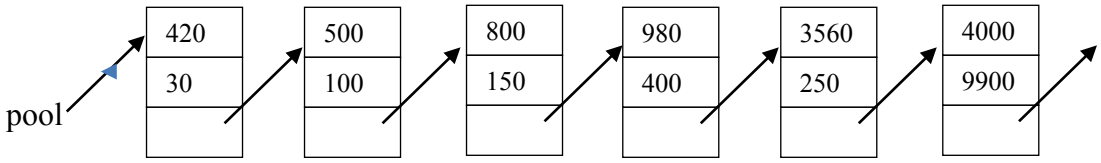


Figure1: Pool of blocks

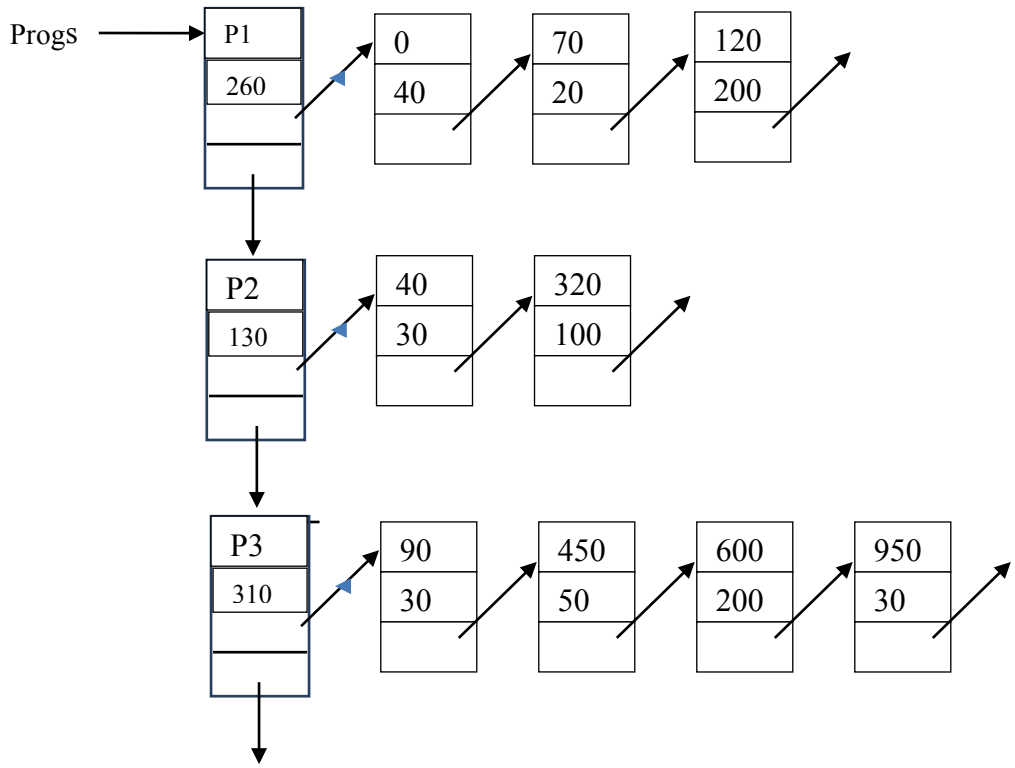


Figure2: Programs list containing three Programs P1, P2 and P3

In order to implement the memory management system, you need to implement the following classes

Class Node

Data members:

- data
- next

Class template of a singly linked list with nested node and Dummy head/tail

Data member:

- head
- tail
- size

Class of block

Data member:

- start_byte_ID
- total_bytes

Class of program

Data members:

- Id
- Size (memory)
- Link list of blocks

Class MemoryManagementSystem
Data members:

- pool (Sorted Linked list of block)
- Progs (Linked list of Programs)
- sizeOfMemory
- strategy (a Boolean variable)

Member functions:
GetMem: The GetMem function must take the program Id and size of the memory required. as parameters and should work as follows:

Find the memory of the required size from the Pool of free memory using a strategy (will be explained later). If the requested memory is not available, then return false. Otherwise, Id is searched in the list of programs. If the program Id already exists, then a new block of memory is removed from the Pool of free memory and added to the existing program at the end of the program linked list of blocks. If no such program already exists, then a new program is inserted at the end of Progs and then this block is added to the list of blocks of the newly inserted program.

The memory management system implements one of the **two strategies: First, fit and best fit.** In the first fit strategy, the memory management system allocates the first available block of memory that fits the program requirement. But in the best fit strategy, the management system allocates the free block of free memory that has the smallest size and meets the program's requirement.
For example, a new program P4 requests a block of 200 size. According to the first fit strategy, the memory will be allocated starting at 980 byte. This block has 400 bytes. The first 200 bytes will be allocated to P4 and remaining 200 resides in the Pool. The updated pool and Progs list is shown the figure 3 and figure 4

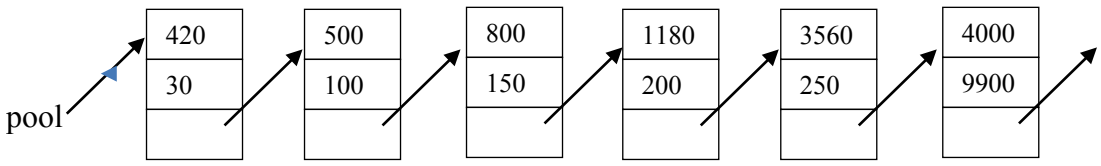


Figure3: Updated Pool of blocks using first fit strategy

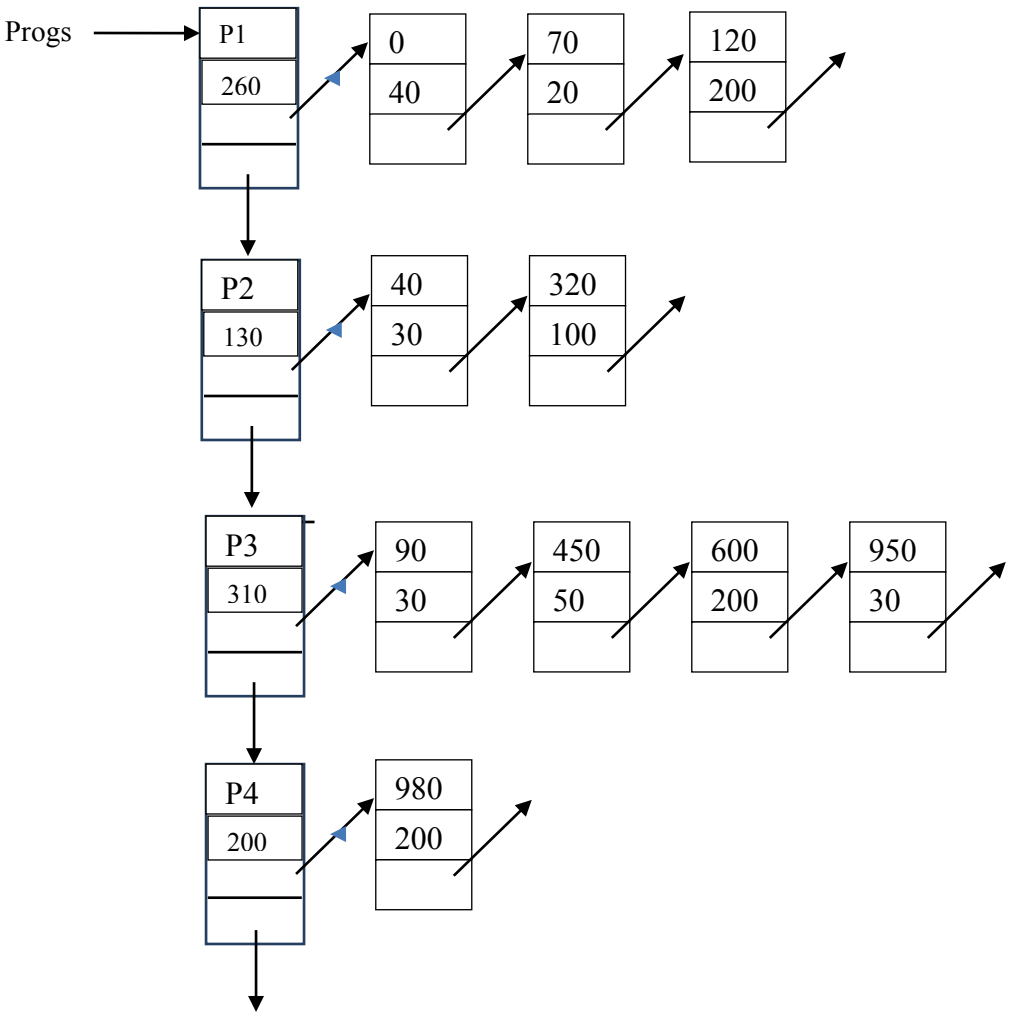


Figure4: Updated Progs list using first fit Strategy

However, if the Strategy is the best fit, then the block starting at 3560 bytes will be allocated. The updated Pool and Progs is shown in Figure 5 and 6.

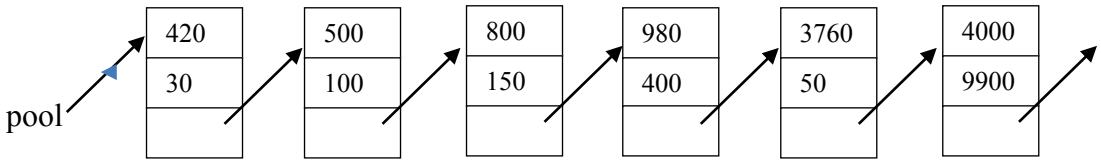


Figure5: Updated Pool of blocks using best fit strategy

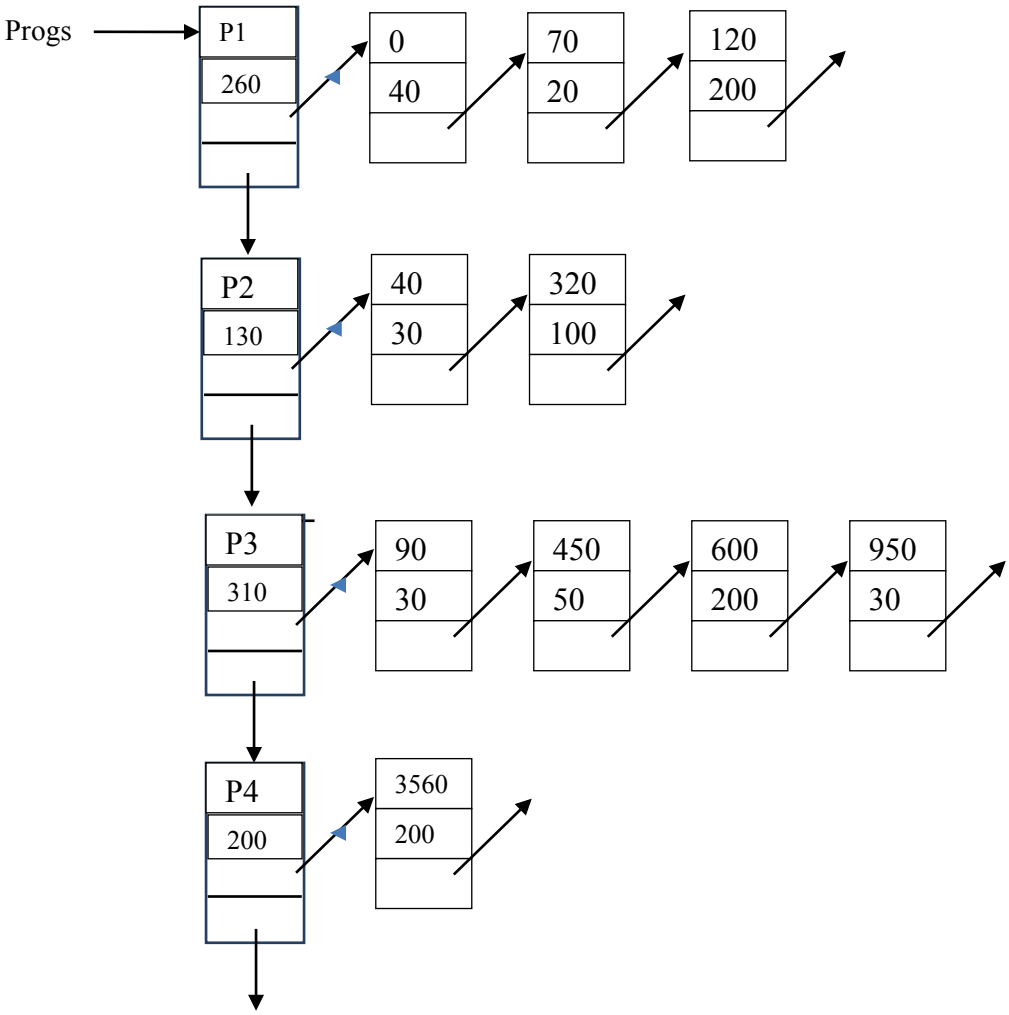


Figure6: Updated Progs list using best fit strategy

DeleteProgram:

This function must take the program Id as a parameter and delete the program with the given Id from Progs. Also, move all the blocks of the memory to pool in sorted order. If two consecutive block become one single contiguous block, then you must also merge them into one single block. For example, if program P3 is deleted, the block starting at 450 must be merged with the block in Pool starting at byte position 420. The updated Pool will be as follows

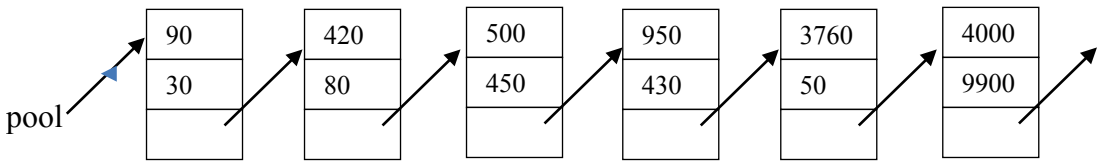


Figure5: Updated Pool of blocks after deletion of P3

Constructor:
Initially, Progs is empty and Pool has only one block. SizeofMemory and Strategy will be initialized to user provided values and will remain the same throughout the program.

Provide a driver function that creates a memory management system object and a menu that prompts the user to perform different operations of the memory management.

VERY IMPORTANT

- Academic integrity is expected of all students. Plagiarism or cheating in any assessment will result in negative markings or an **F** grade in the course and possibly more severe penalties.
- Please make sure that your implementation of this task is efficient and follows all the principles of object-oriented paradigm.