Contiguous Memory Allocation

you may only use the standard C++ header files iostream, list, iomanip, cstdlib, and cstdio. Your submission should compile without errors or warnings using

g++ -Wall -Wextra -Werror -pedantic -std=c++17 MemoryManager.cpp

on the command line.

1 Programming Statement

Write a simple memory manager for dynamically allocated memory. The memory manager will make use of a linked list to keep track of allocated and unallocated blocks of memory;

2 Public interface

The public interface for the memory manager must be as follows (it assumes that the file iostream has been included): MemoryManager.h

2.1 Member functions

MemoryManager — create a memory manager instance of a specified heap size (in bytes). Your memory manager should use new (or malloc) only once to create the entire heap of memory that it will use for memory allocation (note: if you decide to make your own linked list, you may also use new (or malloc) to create new nodes for the list). The construction parameter total\_bytes gives the total size of the heap in bytes. For example, the code fragment

MemoryManager mm(1<<20);

creates an instance of the memory manager, named mm, with a heap size of 1 MB. Each subsequent call to the allocate member function will return a pointer to a chunk of memory within this 1 Mb block.

~MemoryManger — destroys a memory manager instance. As a bare minimum, the destructor will deallocate the heap created by the memory manager constructor. This should be done with a single delete (or free if you used malloc to allocate the heap). Again, if you are making your own linked list, you may invoke delete (or free) for the nodes in the list that you remove).

allocate — get a block of memory of a specified size (in bytes) from the heap. On success, the function returns a pointer to a memory block of the requested size; on failure, a zero (null) pointer is returned. Note that the pointer should be recast to a pointer of the desired type. For example, the code fragment

int \*array = (int\*)mm.allocate(100\*sizeof(int));

will allocate space on the heap for an array of 100 integers.

deallocate — frees a block of memory (that was allocated using allocate) from the heap. The value of pointer should be a non-zero pointer that was returned by the allocate member function. The code fragment

mm.deallocate(array);

frees the space on the heap used by the array of integers allocated above. If pointer has a value of zero, refers to an address that was not returned by allocate, or refers to the addres of a block that was previously deallocated, then the effect of deallocate is undefined; no error checking is performed.

dump — prints information about the current structure of the heap to the specified output stream. This function is intended for debugging purposes; it prints a list of the current blocks on the heap: the starting address of the block, the block size (in bytes), and whether the block is currently in use (allocated) or not (deallocated/free). The code fragment

mm.dump(std::cout);

prints infomation about block structure of the heap to the standard output. An example of the ouput from this function might be something like this

relative to start address: 804c008

byte count: 78

allocated? false

relative to start address: 804c080

byte count: 28

allocated? true

relative to start address: 804c0a8

byte count: f60

allocated? false

which indicates that the heap is currently divided into three blocks of size 78h, 28h, and F60h bytes, respectively; only the middle block is allocated, the other two are free. Your code should emulate this output as much as possible; in particular, all numerical values should be in hexadecimal.

2.2 Details on allocate and deallocate

The allocate member function should search for a block of memory on the heap of the requested size. You should choose the first fit for the block.

Not only should the deallocate function mark the specified block as free (deallocated), but should also consolidate adjacent free blocks into a single free block. For example, if the heap currently has the structure indicated above in the description of the dump function, then the call

mm.deallocate(0x804c080);

would result in the heap having the structure

start address: 804c008

byte count: 1000

allocated? false

i.e., all three of the blocks on the heap are consolidated into a single free block.