Download and Extract

An initial setup of files is provided to you via a shell script: Download potd-q24

Using a terminal, extract the initial files by running the shell script you just downloaded (you will need to navigate to the directory where you saved the file):

```
sh potd-q24.sh
```

Your files for this problem will be in the `potd-q24` directory.

Simulating Memory

We are going to simulate a memory structure with a doubly linked list. Each node represents a chunk of memory. Adjacent nodes imply that the corresponding chunks of memory are also adjacent to each other.

Nodes contain 3 things:

• An **address** representing where this memory chunk starts: It will be in the range [0x0000, 0x10000]

• A **boolean flag** to indicate whether it is in use or not

• **Pointers** to previous and next memory chunks (may be `NULL` if the chunks are the ends of the memory)

Note: we will not be concerning ourselves with the contents of these memory chunks.

Fragmentation

Consider the following scenario:

You have 64 bytes of memory. You allocate 4 chunks of memory, each 16 bytes. As such they are all adjacent to each other and they fill up your entire memory. You free the first chunk, and then the third chunk. If you now try to allocate 32 bytes of memory, your program might say that is impossible because your free memory is `fragmented`, none of the free chunks are at least 32 bytes in size, even though you do have 32 bytes of free memory in total. The solution is to `defragment` your memory: merge all of your free chunks of memory into one big chunk.

What if you free the second chunk and then the third chunk? Well the allocation should pass because when you free the third chunk, you see that 2 chunks of free memory are adjacent to each other, and so you `merge` them automatically. This creates 1 chunk of 32 bytes of memory, which allows the allocation to pass.

Your Job | TL;DR

Your job is to implement the functions `free` and `defragment`.

`free` takes in the address of a memory chunk and marks it as free. It also merges chunks of adjacent free memory when appropriate.

`defragment` should reorganize your memory structure so that all of the allocated memory is grouped together [starting at `0x0000`] and you have one large chunk of free memory at the end.

POTD 24

Total points:	0/1
Score:	0%

Question

Value:	1
History:	
Awarded points:	0/1

Report an error in this question

Previous question

Next question

Upload Solution

Drop files here or click to upload.

Only the files listed below will be accepted—others will be ignored.

Files

○ Memory.cpp

not uploaded

Save & Grade

Save only