P2 FAQS

public MemoryManager(unsigned wordSize, std::function allocator) Constructor;

sets native word size (in bytes, for alignment) and default allocator for finding a memory hole.

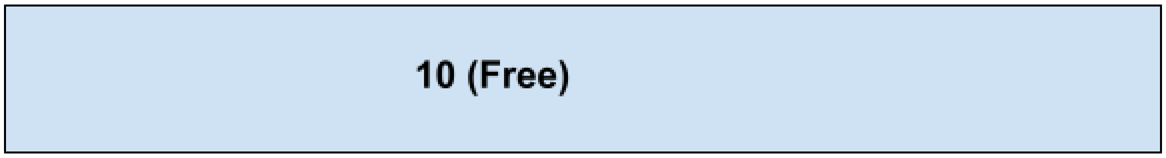
The word size is the number of bytes in a word. There's some conversion that occurs in this project so it's important to know that from smallest to biggest we have:
Bits -> Bytes -> Words -> Blocks

MemoryManager(8, bestFit) would set the wordSize to 8 bytes and the allocating algorithm to bestFit

public void initialize(size_t sizeInWords)

Instantiates block of requested size, no larger than 65536 words; cleans up previous block if applicable.

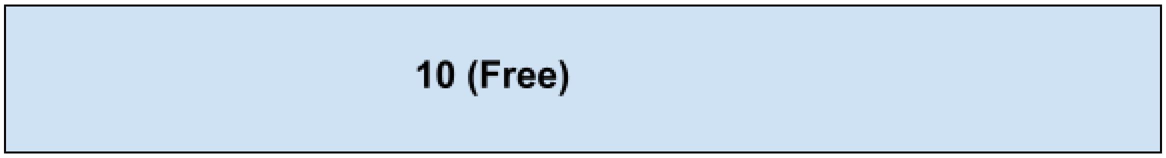
Initialize(10)



This is the initialized block of free memory. All allocated memory will be stored in this block (if there is room). In this case we initialized the block to have a capacity in words of 10 words. The capacity in bytes would be wordSize*sizeInWords

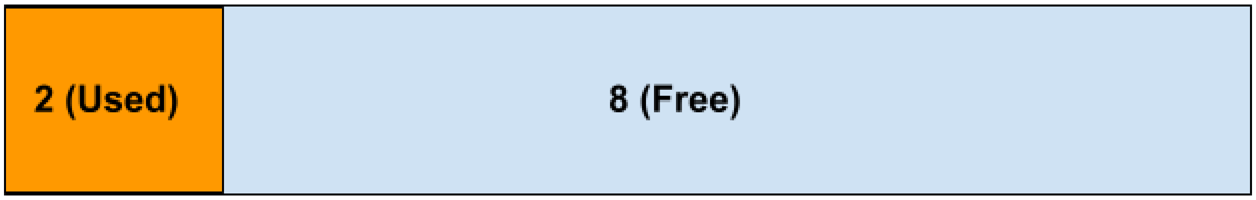
public void *allocate(size_t sizeInBytes)

Allocates a memory using the allocator function. If no memory is available or size is invalid, returns nullptr



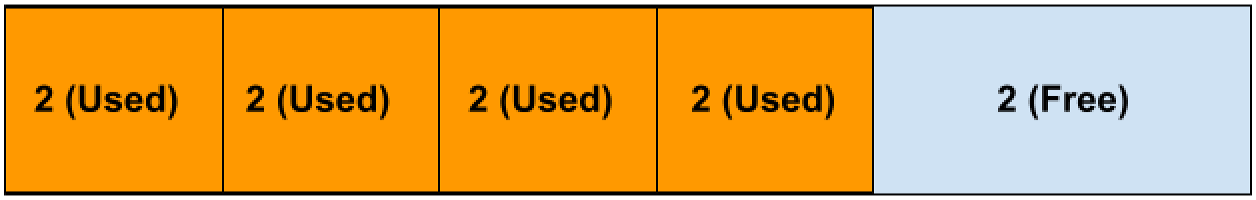
When we initialized, we created a free block of 10 words

Allocate(16)



When allocate is called convert the byte size to words. Then call the allocation algorithm to decide where to store the memory within the block. This splits the whole block into two blocks, a used block and a free block.

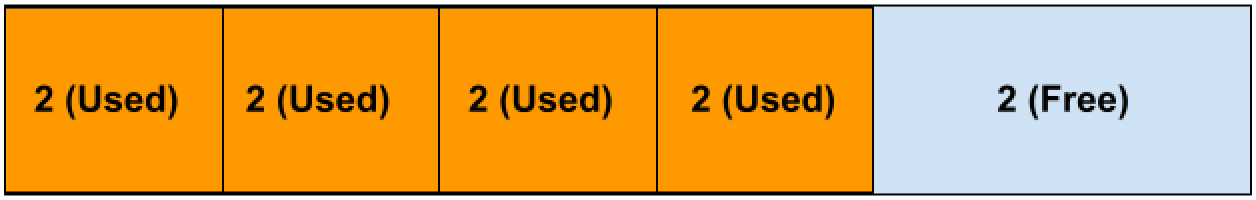
Allocate(16)
Allocate(16)
Allocate(16)



Allocating 16 bytes three more times would result in 4 used blocks and 1 free block. Altogether, they still have a total size of 10 words

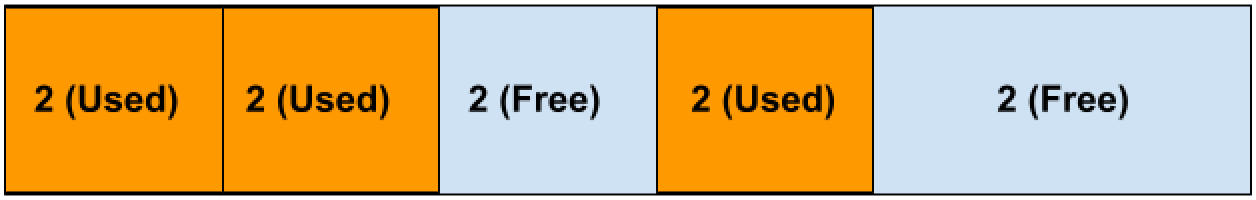
public void free(void *address)

Frees the memory block within the memory manager so that it can be reused.



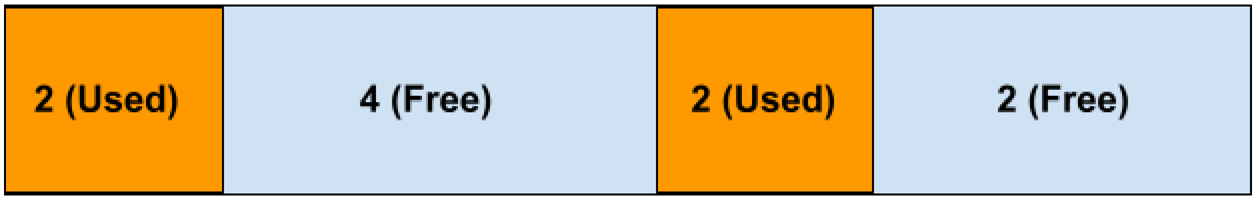
The memory block currently looks like this

Free(*MemoryBlockAddress2)*



This freed MemoryBlock2 which created a new hole.

Free(*MemoryBlockAddress1)*

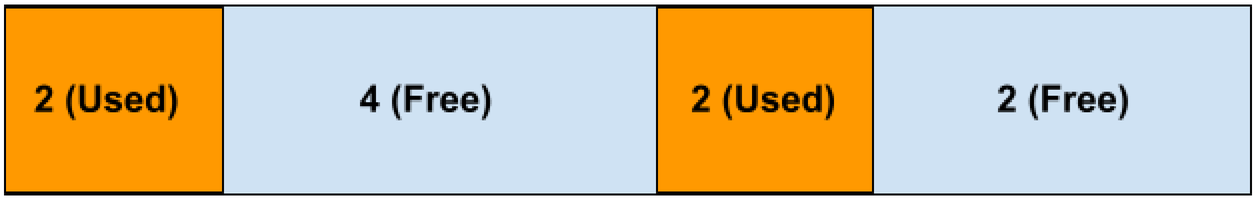


This freed memory block 1. Since the holes were contiguous, they were compacted into one large hole of 4 words.

public void *getList()

Returns a list of information (**in decimal**) about holes for use by the allocator function. Offset and length are in words. If no memory has been allocated, the function should return a NULL pointer.

getList()

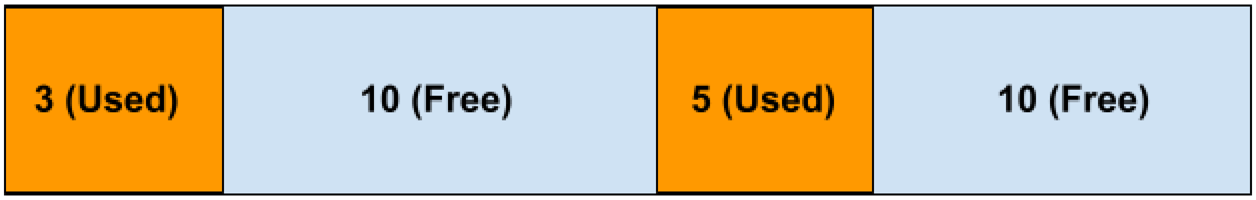


In general we return: [numberOfHoles, offset1, holeSize1, offset2, holeSize2, etc.…]
In this case we would return [2, 2, 4, 8, 2]

public void *getBitmap()

Returns a bit-stream of bits representing whether words are used (1) or free (0). The first two bytes are the size of the bitmap (little-Endian); the rest is the bitmap, word-wise

Assume our block looks like this and we call **getBitMap()**



The bitMap in Little Endian is: 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
Separate it into bytes: 0 0 0 0 | 0 0 0 0 0 0 1 1 | 1 1 1 0 0 0 0 0 | 0 0 0 0 0 1 1 1
Convert to Decimal: 0 | 3 | 224 | 7
The size of the bitMap is 4 bytes.
Therefore we would return (**in decimal**): [4, 0, 7, 224, 3, 0]

Helpful Links:

General
http://www.idc-online.com/technical_references/pdfs/information_technology/Memory_Management_with_Bitmaps_and_Linked_List.pdf

https://opendsa-server.cs.vt.edu/embed/firstFitAV

C++ Help
https://cppsecrets.com/users/687811710010511657495264103109971051084699111109/C00-Functional-Introduction.php

https://stackoverflow.com/questions/6874346/how-do-you-define-a-global-function-in-c

https://stackoverflow.com/questions/11626786/what-does-void-mean-and-how-to-use-it#:~:text=a%20void*%20is%20a%20pointer%2C%20but%20the%20type%20of%20what,the%20function%20to%20use%20it

POSIX Help
(NOTE POSIX calls can only be done in terminal, VScode does not support it by default. Therefore, save dumpMemoryMap for the end)
https://linux.die.net/man/3/open

https://man7.org/linux/man-pages/man2/open.2.html

https://man7.org/linux/man-pages/man2/write.2.html

https://man7.org/linux/man-pages/man2/close.2.html

https://www.howtogeek.com/248780/how-to-compress-and-extract-files-using-the-tar-command-on-linux/

ENCOURAGE MINT