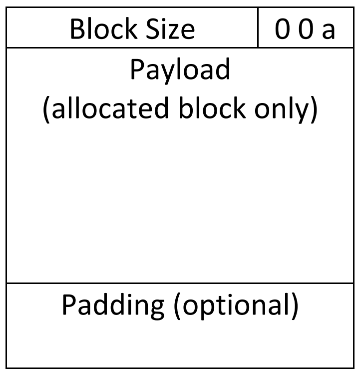
CS5541-Computer Systems
Memory Allocation

Determine the block sizes and header values that would result from the following sequence of malloc requests. Assumptions: (1) The allocator maintains double-word alignment and uses an implicit free list with the block format from the figure below. (2) Block sizes are rounded up to the nearest multiple of 8 bytes.

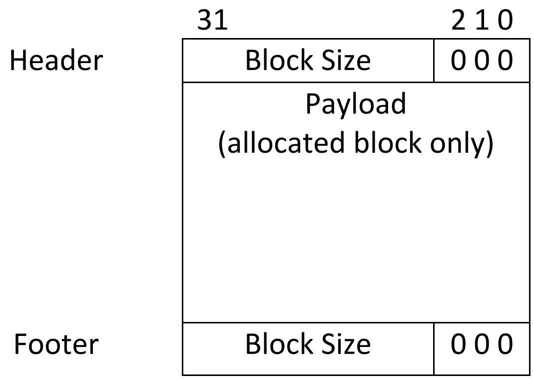


Request	**Block Size (# of Bytes)**	**Block Header (hex)**
malloc(3)		
malloc(11)		
malloc(20)		
malloc(21)		

Determine the minimum block size for each of the following combinations of alignment requirements and block formats. Assumptions: Implicit free list, zero-size payloads are not allowed, and headers and footers are stored in 4-byte words.

Alignment	**Allocated Block**	**Free Block**	**Minimum Block Size (bytes)**
Single Word	Header and Footer	Header and Footer	
Single Word	Header, but no Footer	Header and Footer	
Double Word	Header and Footer	Header and Footer	
Double Word	Header, but no Footer	Header and Footer	

The following problem concerns dynamic storage allocation. Consider an allocator that uses an implicit free list. The layout of each allocated and free memory block is as follows:



Each memory block, either allocated or free, has a size that is a multiple of eight bytes. Thus, only the 29 higher order bits in the header and footer are needed to record block size, which includes the header and footer. The usage of the remaining 3 lower order bits is as follows:
-- bit 0 indicates the use of the current block: 1 for allocated, 0 for free.
-- bit 1 indicates the use of the previous adjacent block: 1 for allocated, 0 for free.
-- bit 2 is unused and is always set to be 0.

Given the address of a word (4 bytes) in the heap in the left column, and the original contents of the heap in the middle column, show the new contents of the heap in the right column after a call to **free(0x400b010)** is executed. Your answers should be given as hex values. Note that the address grows from bottom up. Assume that the allocator uses immediate coalescing, that is, adjacent free blocks are merged immediately each time a block is freed.

Address	Current Value (hex)	New Value (hex)
0x400b028	0x00000012	
0x400b024	0x400b611c	0x400b611c
0x400b020	0x400b512c	0x400b512c
0x400b01c	0x00000012	
0x400b018	0x00000013	
0x400b014	0x400b511c	0x400b511c
0x400b010	0x400b601c	0x400b601c
0x400b00c	0x00000013	
0x400b008	0x00000013	
0x400b004	0x400b601c	0x400b601c
0x400b000	0x400b511c	0x400b511c
0x400affc	0x00000013	

Below you are given a series of memory requests as they might appear in a user's program. The heap is represented as a row of boxes, where each box is a single block on the heap, and the bottom of the heap is the left-most box. Simulate the calls to malloc( ) or free( ) on the left by marking each block in the corresponding row. In each block, you should write the total size (including headers and footers) of the block in bytes and either “f” or “a” to mark it as free or allocated, respectively. For example, the following heap contains an allocated block of size 16, followed by a free block of size 32.

16a	32f			

The calls to malloc( ) and free( ) are cumulative, so each call starts from the row above except the first which starts with an empty heap.

Perform the series of calls to malloc and free using first fit to choose a free block for malloc() and immediate coalescing to merge blocks after free().

ptr1 = malloc(32)

ptr2 = malloc(16);

ptr3 = malloc(16);

ptr4 = malloc(40);

free(ptr3);

free(ptr1);

ptr5 = malloc(16);

free(ptr4);

ptr6 = malloc(48);

free(ptr2);

Perform the series of calls to malloc() and free() using best fit to choose a free block for malloc() and immediate coalescing to merge blocks after free().

ptr1 = malloc(32)

ptr2 = malloc(16);

ptr3 = malloc(16);

ptr4 = malloc(40);

free(ptr3);

free(ptr1);

ptr5 = malloc(16);

free(ptr4);

ptr6 = malloc(48);

free(ptr2);