Problem 3 of 8

Malloc

Let HEAP_BLOCKS be the total number of blocks in the heap, and FREE_BLOCKS be the total number of blocks in the free list. In an **implicit list** memory allocator:

1) What is the worst-case time complexity (Big-O) of the **first-fit** allocation policy?

○ O(HEAP_BLOCKS)
○ O(FREE_BLOCKS)
○ O(1)

2) What is the worst-case time complexity (Big-O) of the **best-fit** allocation policy?

○ O(HEAP_BLOCKS)
○ O(FREE_BLOCKS)
○ O(1)

In an **explicit list** memory allocator:

3) What is the worst-case time complexity (Big-O) of the **first-fit** allocation policy?

○ O(HEAP_BLOCKS)
○ O(FREE_BLOCKS)
○ O(1)

4) What is the worst-case time complexity (Big-O) of the **best-fit** allocation policy?

○ O(HEAP_BLOCKS)
○ O(FREE_BLOCKS)
● O(1)

Consider the following code:

```
#define TOTAL_POINTERS 16
#include <stdbool.h>

bool jack_biggs_best_jack()
{
    void *pointers[TOTAL_POINTERS];
    int index;

    // First malloc loop
    for (index = 0; index < TOTAL_POINTERS; index++)
    {
        if (index < TOTAL_POINTERS / 2) pointers[index] = malloc(64);
        else pointers[index] = malloc(32);
    }

    // First free loop
    for (index = 0; index < TOTAL_POINTERS / 2; index++)
    {
        free(pointers[2*index]);
    }

    // Second malloc loop
    for (index = 0; index < TOTAL_POINTERS / 4; index++)
    {
        pointers[2*index] = malloc(16);
    }
    // Third malloc loop
    for (index = TOTAL_POINTERS / 4; index < TOTAL_POINTERS / 2; index++)
    {
        pointers[2*index] = malloc(48);
    }

    return true;
}
```

Consider a explicit memory allocator with the following specifications:

- It uses 8-byte header and 8-byte footer.
- The payload is 16-byte aligned.
- The minimum block size is 32 bytes.
- Free blocks are coalesced immediately.
- Splitting is performed only if the resulting block size is greater than or equal to the minimum block size.

Note the following important points for the question:

- Assume that the prologue and epilogue blocks are present, and that the first payload will be appropriately aligned. **Exclude the prologue and epilogue blocks from all your calculations below.**
- The heap is initially empty (0 bytes).
- The memory allocator uses a **first-fit** allocation policy and **FIFO** insertion policy.

Please simplify all your answers to an integer or the simplest fraction.

5) How many total bytes are 'sbrk'ed as a result of the first malloc loop?

6) How many coalesces are performed on the first free loop?

7) What is the total number of bytes requested by the user by the end of the trace? Exclude bytes that have been freed.

8) How much memory in total is 'sbrk'ed at the end of the trace?

9) How many of those bytes are currently free? Include headers and footers of the free blocks.

10) How many bytes are lost to internal fragmentation? Internal fragmentation is all non-payload bytes in allocated blocks.