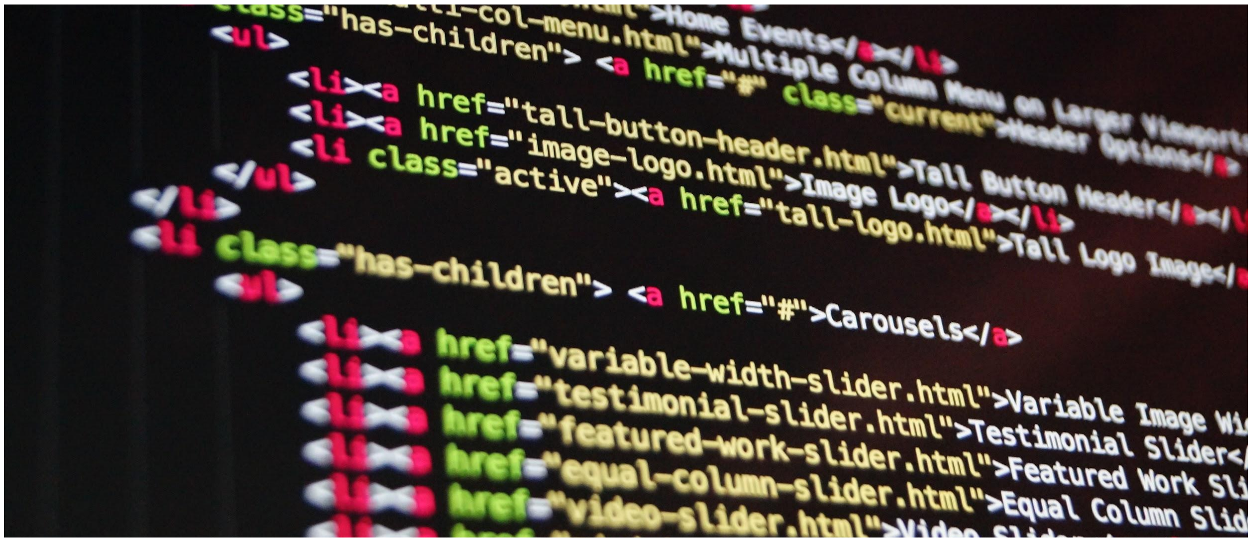


DYNAMIC MEMORY ALLOCATION REPORT

MALLOC, REALLOC, CALLOC, AND FREE

HARITA AGARWAL
4/27/2023

How to run the program?

- First log in into your VM with ssh <username>@172.18.233.81/82/83/84
- Go to the respective directory if applicable (cd assignment3)
- Once you are inside the directory, type make and then ./myprogram for modified and ./myprogramOG for original.

Memory Leak Calculation

In the above code, the memoryLeaks function in functions.c is meant to determine the total amount of memory that the program has allocated but has not yet released.

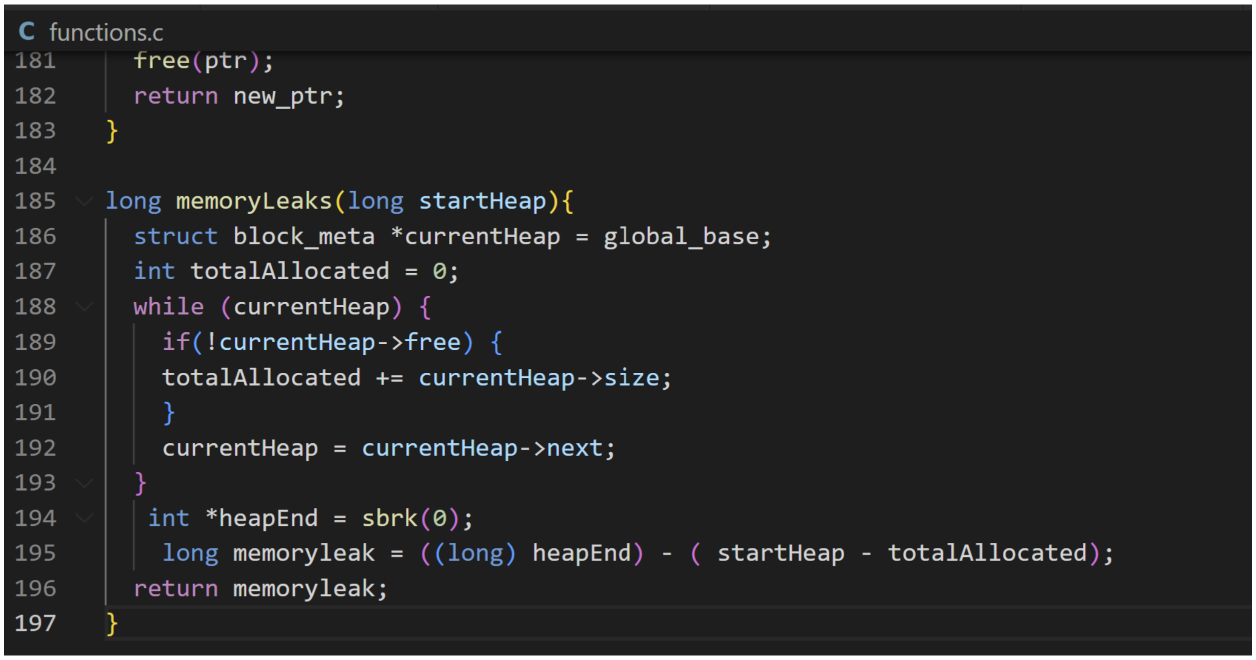
The function takes a heap beginning address (startHeap) as an argument, which is the program's initial value when it started. It sets a pointer currentHeap to the memory block structure's global base. It also sets the totalAllocated counter to 0, which will be used to collect the total amount of memory allocated but not removed.

The function then runs through each memory block in the linked list, which is kept going by the malloc and free operations, until the list is finished. It examines each block to see if it is currently in use (not marked as "free"). If the block is in use, its size is added to “totalAllocated”.

After the loop is finished, the function uses the sbrk function with an argument of zero to get the current program break (heapEnd). The memory leak is then calculated by removing the beginning heap address from the current program break and adding back the entire amount of memory allocated but not released.

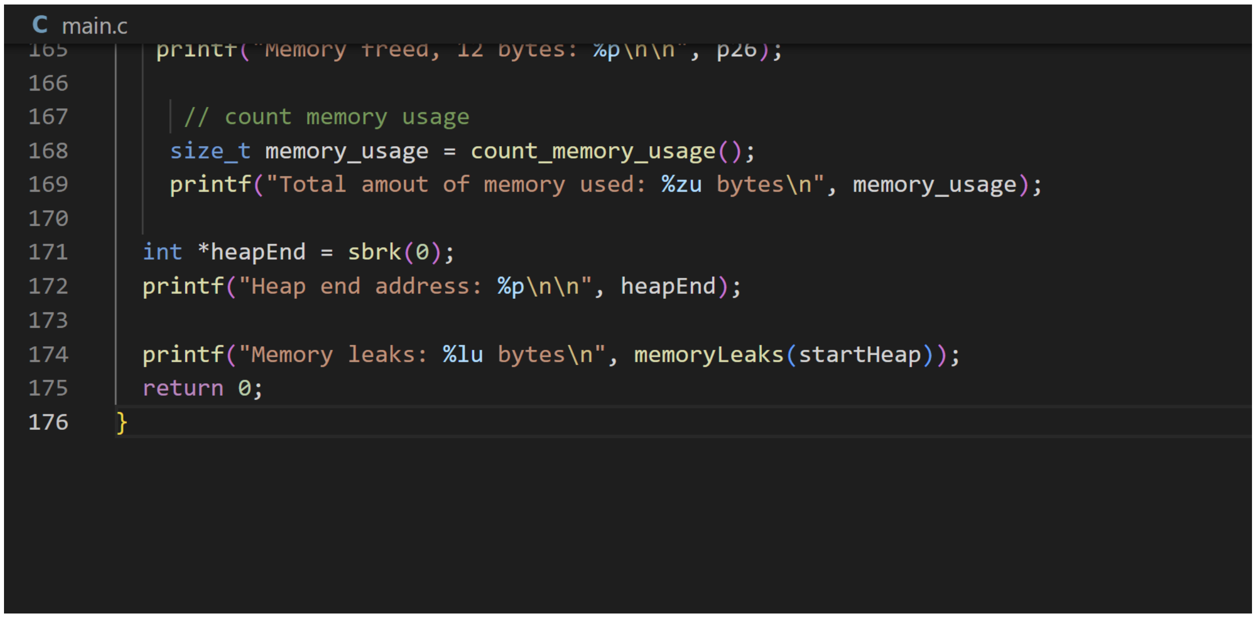
The memoryLeaks function is called in main.c using the initial heap address (startHeap) as an argument. Using printf, the resulting memory leak is printed.

Code:



functions.c

```c
    free(ptr);
    return new_ptr;
}

long memoryLeaks(long startHeap){
    struct block_meta *currentHeap = global_base;
    int totalAllocated = 0;
    while (currentHeap) {
        if(!currentHeap->free) {
        totalAllocated += currentHeap->size;
        }
        currentHeap = currentHeap->next;
    }
     int *heapEnd = sbrk(0);
      long memoryleak = ((long) heapEnd) - ( startHeap - totalAllocated);
    return memoryleak;
}
```

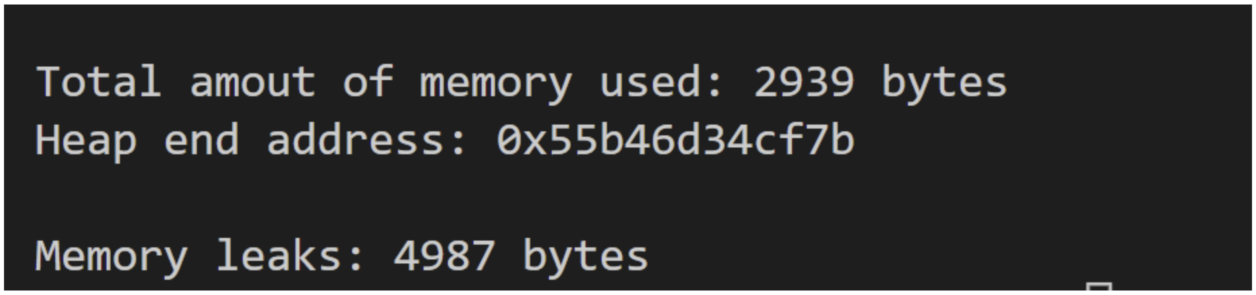


main.c

```c
    printf("Memory freed, 12 bytes: %p\n\n", p26);

     // count memory usage
     size_t memory_usage = count_memory_usage();
     printf("Total amout of memory used: %zu bytes\n", memory_usage);

    int *heapEnd = sbrk(0);
    printf("Heap end address: %p\n\n", heapEnd);

    printf("Memory leaks: %lu bytes\n", memoryLeaks(startHeap));
    return 0;
}
```

Output:

This is the total number of memory allocated and memory leak with all memory implementations.



```
Total amout of memory used: 2939 bytes
Heap end address: 0x55b46d34cf7b

Memory leaks: 4987 bytes
```

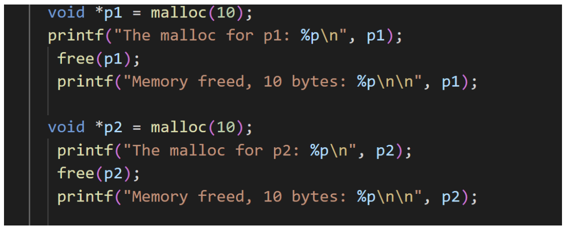
Function Modification:

- In functions.c, "find_free_block" method was modified to utilize the best fit implementation.
- In functions.c, "block_meta" struct was modified to represent a node in a linked list by adding a "prev" variable.
- In functions.c, "request_space" method was modified to set the "prev" pointer to the last node in the list.
- In functions.c, "count_memory_usage" was modified to calculate the total amount of memory used by currently allocated blocks in the doubly linked list.
- In functions.c, implementation of memory leak checker ensures there is no memory leak and all memory that was allocated is freed. My logic behind the calculations of memory leak is mentioned above in detail.
- In main.c, "memoryLeaks" is called after defined in the functions.c.

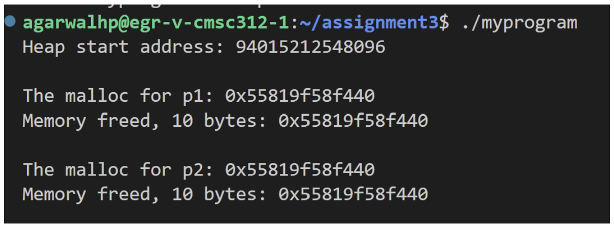
Different outputs

I am sharing a few of the instances from the code but refer to my entire code using the instruction of how to run it and see the complete outcome.

The output for p1 and p2 will be the same since the memory addresses returned by the malloc() function both contain 10 bytes so both have the same memory address for p1 and p2.



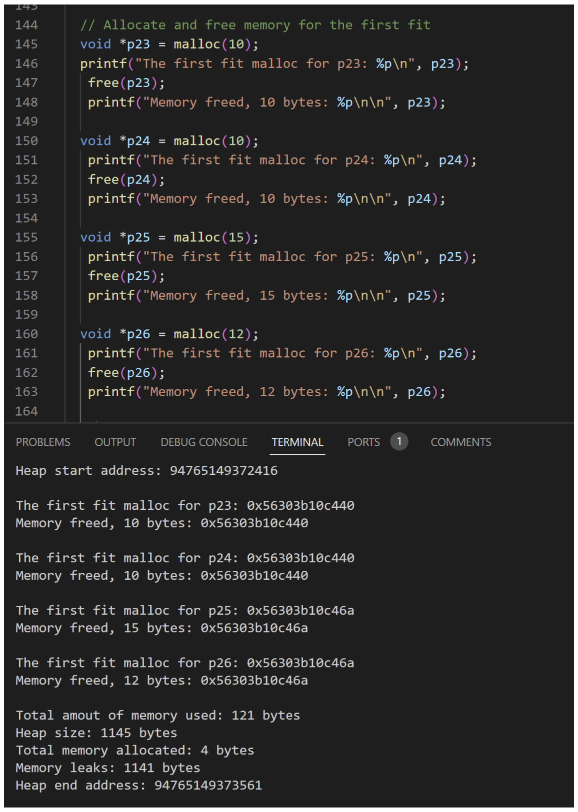
```c
void *p1 = malloc(10);
printf("The malloc for p1: %p\n", p1);
 free(p1);
 printf("Memory freed, 10 bytes: %p\n\n", p1);

void *p2 = malloc(10);
 printf("The malloc for p2: %p\n", p2);
 free(p2);
 printf("Memory freed, 10 bytes: %p\n\n", p2);
```



```
agarwalhp@egr-v-cmsc312-1:~/assignment3$ ./myprogram
Heap start address: 94015212548096

The malloc for p1: 0x55819f58f440
Memory freed, 10 bytes: 0x55819f58f440

The malloc for p2: 0x55819f58f440
Memory freed, 10 bytes: 0x55819f58f440
```

For memory allocation, this below code uses the first fit method rather than the best fit implementation.

In the first fit algorithm, the allocator searches the heap for a suitable memory block and selects the first block that is large enough to accommodate the requested size. The best fit method, on the other hand, examines the whole heap for the smallest block that is large enough to handle the specified size. Each call to malloc() in the code demands a specified amount of memory. The relevant pointer is assigned the first available block of memory that is large enough to handle the request. As a result, the code employs the first fit approach for memory allocation.



```c
// Allocate and free memory for the first fit
void *p23 = malloc(10);
printf("The first fit malloc for p23: %p\n", p23);
 free(p23);
 printf("Memory freed, 10 bytes: %p\n\n", p23);

void *p24 = malloc(10);
 printf("The first fit malloc for p24: %p\n", p24);
 free(p24);
 printf("Memory freed, 10 bytes: %p\n\n", p24);

void *p25 = malloc(15);
 printf("The first fit malloc for p25: %p\n", p25);
 free(p25);
 printf("Memory freed, 15 bytes: %p\n\n", p25);

void *p26 = malloc(12);
 printf("The first fit malloc for p26: %p\n", p26);
 free(p26);
 printf("Memory freed, 12 bytes: %p\n\n", p26);
```

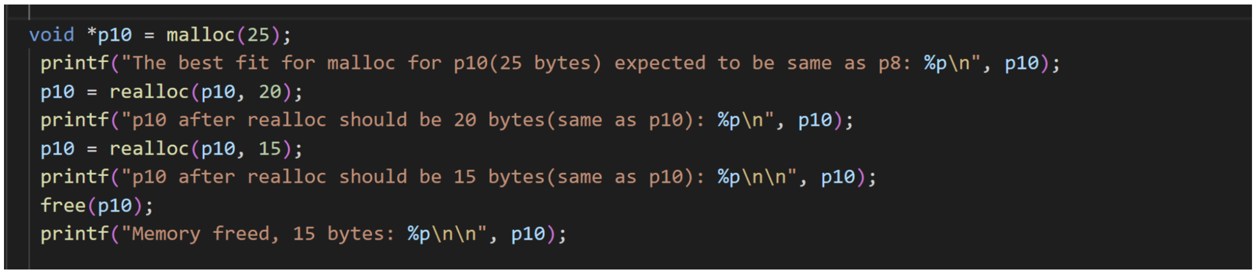
```
Heap start address: 94765149372416

The first fit malloc for p23: 0x56303b10c440
Memory freed, 10 bytes: 0x56303b10c440

The first fit malloc for p24: 0x56303b10c440
Memory freed, 10 bytes: 0x56303b10c440

The first fit malloc for p25: 0x56303b10c46a
Memory freed, 15 bytes: 0x56303b10c46a

The first fit malloc for p26: 0x56303b10c46a
Memory freed, 12 bytes: 0x56303b10c46a

Total amout of memory used: 121 bytes
Heap size: 1145 bytes
Total memory allocated: 4 bytes
Memory leaks: 1141 bytes
Heap end address: 94765149373561
```

In this best fit implementation example, p10 is given 25 bytes of memory and its pointer value is given. The block referenced by p10 is subsequently resized by using realloc()

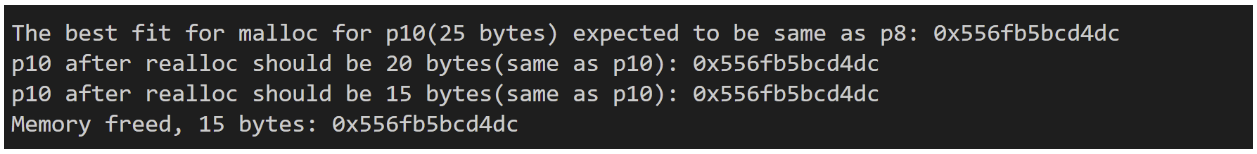
twice. The first call reduces the block to 20 bytes, whereas the second reduces it to 15 bytes. The identical block of memory is returned in both circumstances, and the pointer value of p10 remains unchanged.

The pointer value of p10 is reported again once the memory is released using free(). Since the memory has been freed, the pointer value is now 0x556fb5bcd4dc (represented in the output as (0x556fb5bcd4dc)).

The code's output may differ based on the available free memory blocks and the precise implementation of malloc(), realloc(), and free().

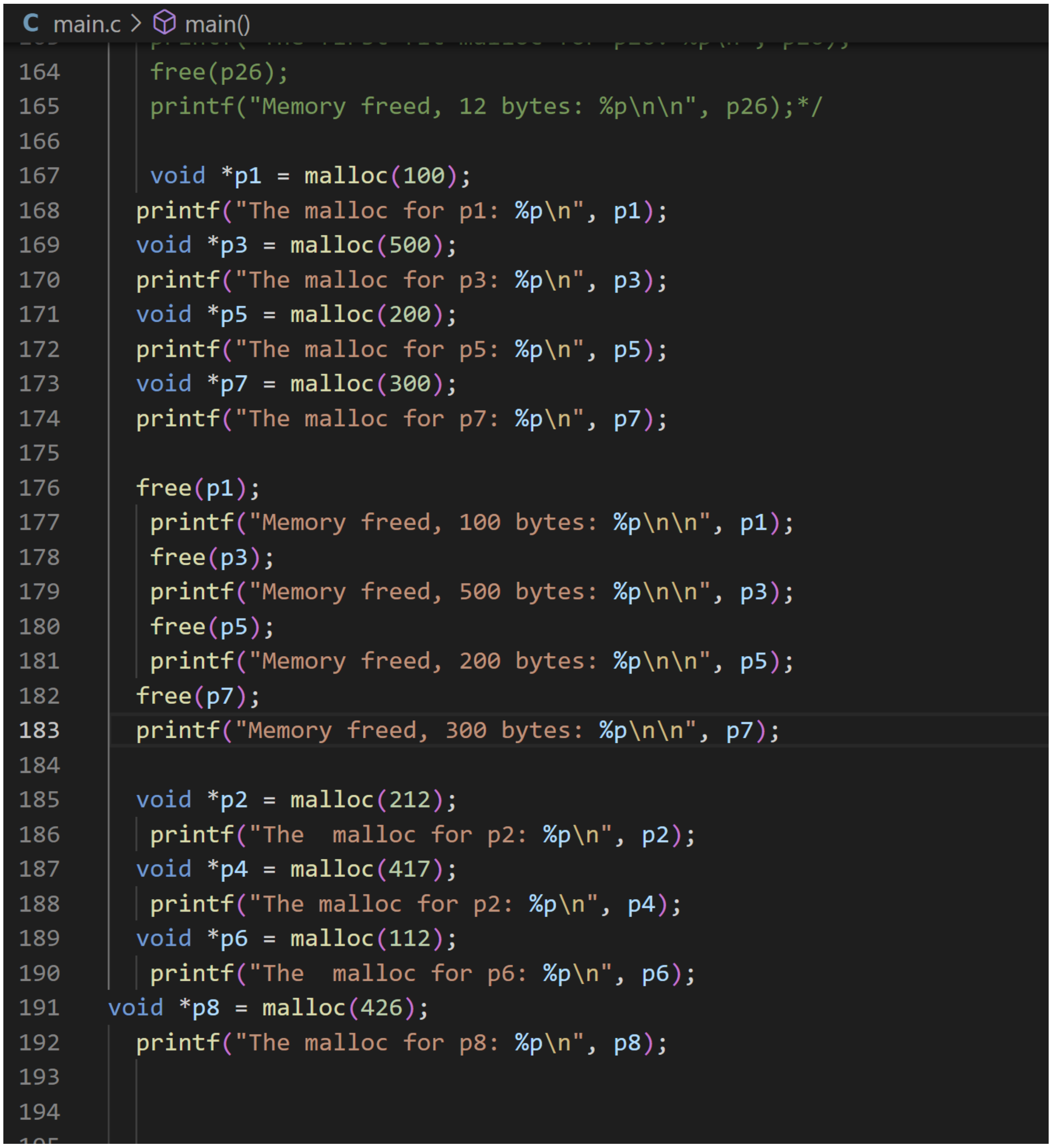


```
void *p10 = malloc(25);
 printf("The best fit for malloc for p10(25 bytes) expected to be same as p8: %p\n", p10);
 p10 = realloc(p10, 20);
 printf("p10 after realloc should be 20 bytes(same as p10): %p\n", p10);
 p10 = realloc(p10, 15);
 printf("p10 after realloc should be 15 bytes(same as p10): %p\n\n", p10);
 free(p10);
 printf("Memory freed, 15 bytes: %p\n\n", p10);
```

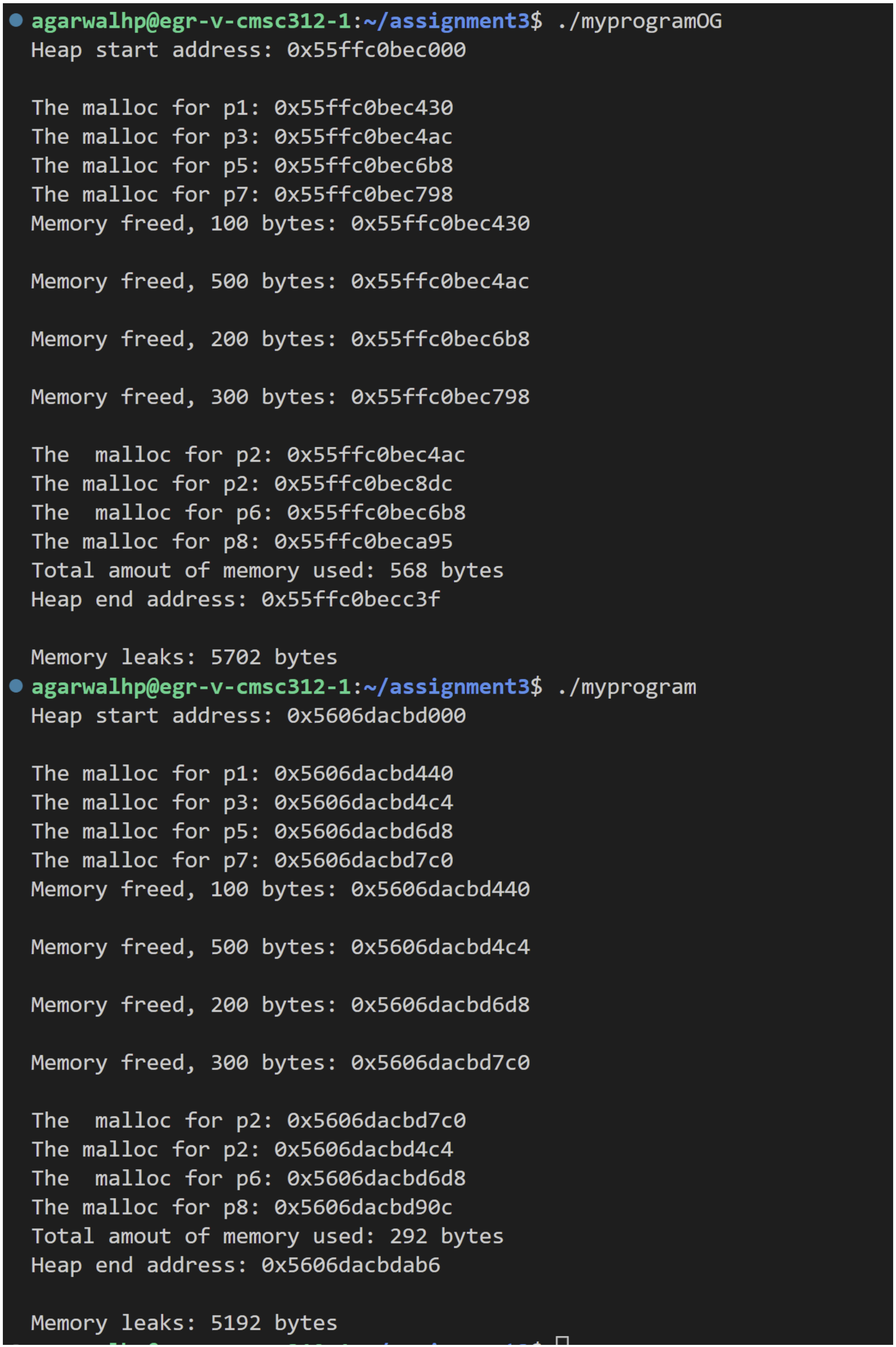


```
The best fit for malloc for p10(25 bytes) expected to be same as p8: 0x556fb5bcd4dc
p10 after realloc should be 20 bytes(same as p10): 0x556fb5bcd4dc
p10 after realloc should be 15 bytes(same as p10): 0x556fb5bcd4dc
Memory freed, 15 bytes: 0x556fb5bcd4dc
```

I believe that first fit is better because the first fit algorithm can use larger blocks of valuable free memory for smaller requests, it needs to create the heap more frequently than the best fit algorithm, consuming more memory. In the original code, single linked list the memory leak was 4875 bytes and for double linked list it was 4987 bytes which makes sense as it is the implementation on a doubly linked list.



```c
     free(p26);
     printf("Memory freed, 12 bytes: %p\n\n", p26);*/

     void *p1 = malloc(100);
    printf("The malloc for p1: %p\n", p1);
    void *p3 = malloc(500);
    printf("The malloc for p3: %p\n", p3);
    void *p5 = malloc(200);
    printf("The malloc for p5: %p\n", p5);
    void *p7 = malloc(300);
    printf("The malloc for p7: %p\n", p7);

    free(p1);
     printf("Memory freed, 100 bytes: %p\n\n", p1);
     free(p3);
     printf("Memory freed, 500 bytes: %p\n\n", p3);
     free(p5);
     printf("Memory freed, 200 bytes: %p\n\n", p5);
    free(p7);
    printf("Memory freed, 300 bytes: %p\n\n", p7);

    void *p2 = malloc(212);
     printf("The  malloc for p2: %p\n", p2);
    void *p4 = malloc(417);
     printf("The malloc for p2: %p\n", p4);
    void *p6 = malloc(112);
     printf("The  malloc for p6: %p\n", p6);
  void *p8 = malloc(426);
    printf("The malloc for p8: %p\n", p8);
```



```
agarwalhp@egr-v-cmsc312-1:~/assignment3$ ./myprogramOG
Heap start address: 0x55ffc0bec000

The malloc for p1: 0x55ffc0bec430
The malloc for p3: 0x55ffc0bec4ac
The malloc for p5: 0x55ffc0bec6b8
The malloc for p7: 0x55ffc0bec798
Memory freed, 100 bytes: 0x55ffc0bec430

Memory freed, 500 bytes: 0x55ffc0bec4ac

Memory freed, 200 bytes: 0x55ffc0bec6b8

Memory freed, 300 bytes: 0x55ffc0bec798

The  malloc for p2: 0x55ffc0bec4ac
The malloc for p2: 0x55ffc0bec8dc
The  malloc for p6: 0x55ffc0bec6b8
The malloc for p8: 0x55ffc0beca95
Total amout of memory used: 568 bytes
Heap end address: 0x55ffc0becc3f

Memory leaks: 5702 bytes
agarwalhp@egr-v-cmsc312-1:~/assignment3$ ./myprogram
Heap start address: 0x5606dacbd000

The malloc for p1: 0x5606dacbd440
The malloc for p3: 0x5606dacbd4c4
The malloc for p5: 0x5606dacbd6d8
The malloc for p7: 0x5606dacbd7c0
Memory freed, 100 bytes: 0x5606dacbd440

Memory freed, 500 bytes: 0x5606dacbd4c4

Memory freed, 200 bytes: 0x5606dacbd6d8

Memory freed, 300 bytes: 0x5606dacbd7c0

The  malloc for p2: 0x5606dacbd7c0
The malloc for p2: 0x5606dacbd4c4
The  malloc for p6: 0x5606dacbd6d8
The malloc for p8: 0x5606dacbd90c
Total amout of memory used: 292 bytes
Heap end address: 0x5606dacbdab6

Memory leaks: 5192 bytes
```

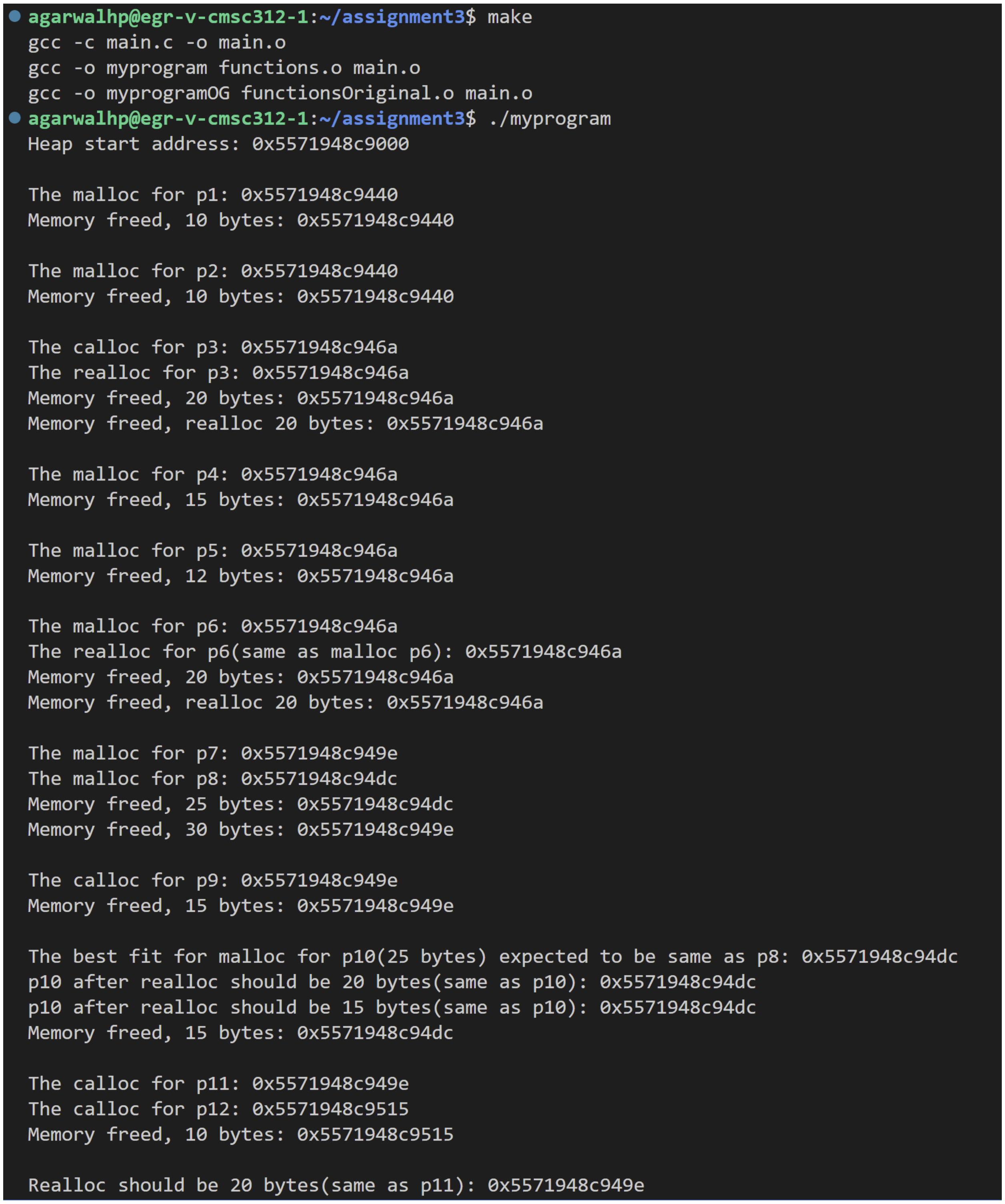
In this case, the best fit is better than the first fit because the modified code allocations are big enough to make up for the overhead of the doubly linked list.

Code exception/condition/notes

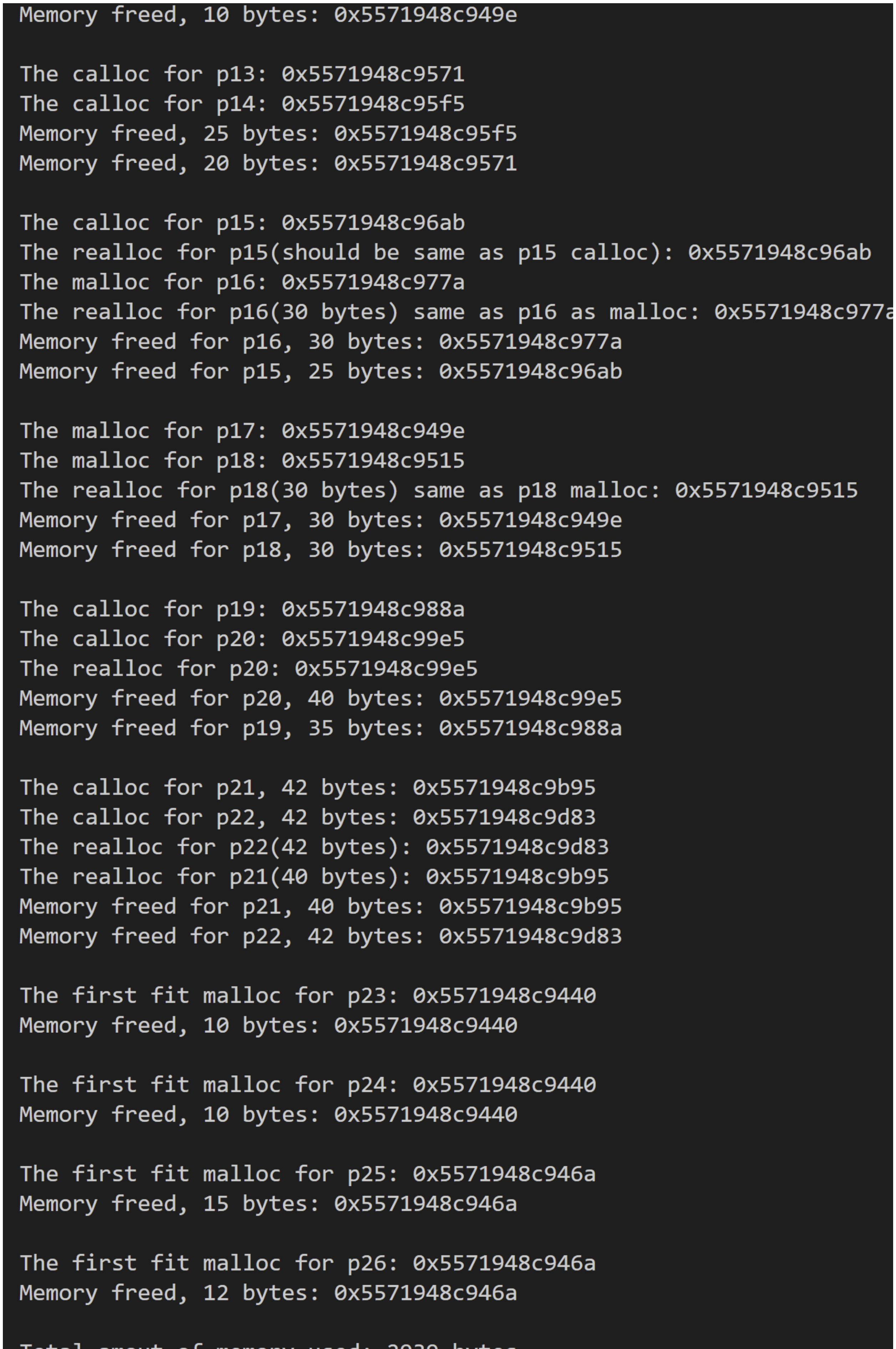
- No exception or condition in the code.
- The print statements were for me to keep track of the code and to make sure each memory allocation was getting freed. Some of the print statements are not meant for first fit implementation; they are just proving best fit implementation or different instances that the code is working.

Full Output:

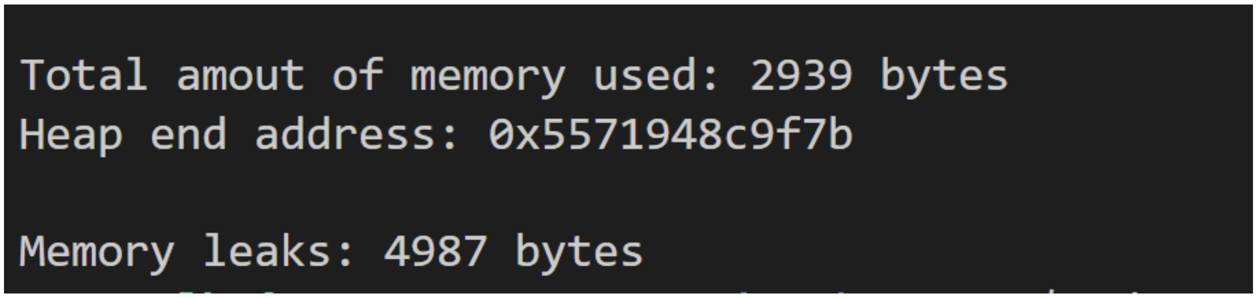
This is the modified version of the code ./myprogram which shows the double linked list implementation.



```
agarwalhp@egr-v-cmsc312-1:~/assignment3$ make
gcc -c main.c -o main.o
gcc -o myprogram functions.o main.o
gcc -o myprogramOG functionsOriginal.o main.o
agarwalhp@egr-v-cmsc312-1:~/assignment3$ ./myprogram
Heap start address: 0x5571948c9000

The malloc for p1: 0x5571948c9440
Memory freed, 10 bytes: 0x5571948c9440

The malloc for p2: 0x5571948c9440
Memory freed, 10 bytes: 0x5571948c9440

The calloc for p3: 0x5571948c946a
The realloc for p3: 0x5571948c946a
Memory freed, 20 bytes: 0x5571948c946a
Memory freed, realloc 20 bytes: 0x5571948c946a

The malloc for p4: 0x5571948c946a
Memory freed, 15 bytes: 0x5571948c946a

The malloc for p5: 0x5571948c946a
Memory freed, 12 bytes: 0x5571948c946a

The malloc for p6: 0x5571948c946a
The realloc for p6(same as malloc p6): 0x5571948c946a
Memory freed, 20 bytes: 0x5571948c946a
Memory freed, realloc 20 bytes: 0x5571948c946a

The malloc for p7: 0x5571948c949e
The malloc for p8: 0x5571948c94dc
Memory freed, 25 bytes: 0x5571948c94dc
Memory freed, 30 bytes: 0x5571948c949e

The calloc for p9: 0x5571948c949e
Memory freed, 15 bytes: 0x5571948c949e

The best fit for malloc for p10(25 bytes) expected to be same as p8: 0x5571948c94dc
p10 after realloc should be 20 bytes(same as p10): 0x5571948c94dc
p10 after realloc should be 15 bytes(same as p10): 0x5571948c94dc
Memory freed, 15 bytes: 0x5571948c94dc

The calloc for p11: 0x5571948c949e
The calloc for p12: 0x5571948c9515
Memory freed, 10 bytes: 0x5571948c9515

Realloc should be 20 bytes(same as p11): 0x5571948c949e
```

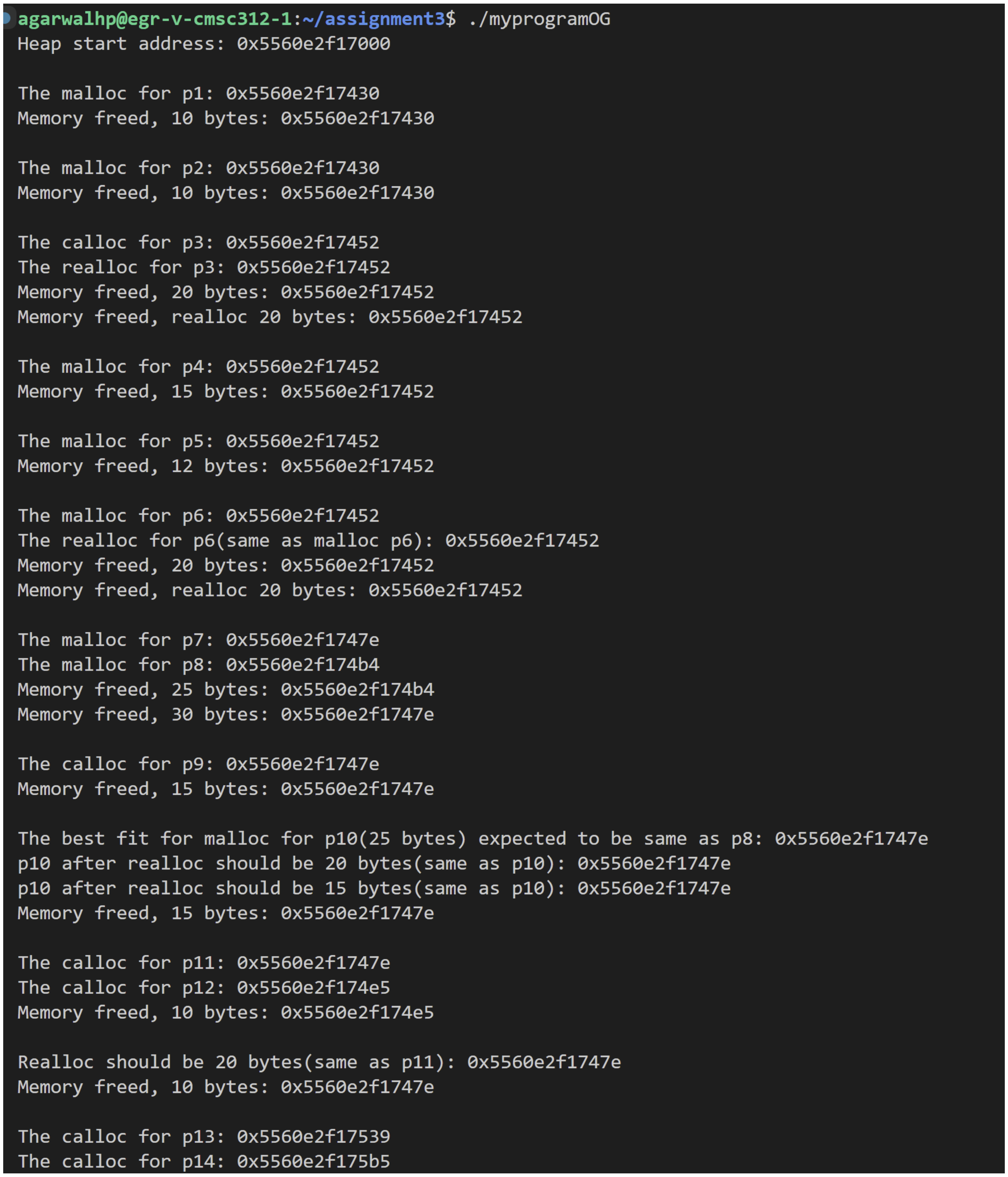


```
Memory freed, 10 bytes: 0x5571948c949e

The calloc for p13: 0x5571948c9571
The calloc for p14: 0x5571948c95f5
Memory freed, 25 bytes: 0x5571948c95f5
Memory freed, 20 bytes: 0x5571948c9571

The calloc for p15: 0x5571948c96ab
The realloc for p15(should be same as p15 calloc): 0x5571948c96ab
The malloc for p16: 0x5571948c977a
The realloc for p16(30 bytes) same as p16 as malloc: 0x5571948c977a
Memory freed for p16, 30 bytes: 0x5571948c977a
Memory freed for p15, 25 bytes: 0x5571948c96ab

The malloc for p17: 0x5571948c949e
The malloc for p18: 0x5571948c9515
The realloc for p18(30 bytes) same as p18 malloc: 0x5571948c9515
Memory freed for p17, 30 bytes: 0x5571948c949e
Memory freed for p18, 30 bytes: 0x5571948c9515

The calloc for p19: 0x5571948c988a
The calloc for p20: 0x5571948c99e5
The realloc for p20: 0x5571948c99e5
Memory freed for p20, 40 bytes: 0x5571948c99e5
Memory freed for p19, 35 bytes: 0x5571948c988a

The calloc for p21, 42 bytes: 0x5571948c9b95
The calloc for p22, 42 bytes: 0x5571948c9d83
The realloc for p22(42 bytes): 0x5571948c9d83
The realloc for p21(40 bytes): 0x5571948c9b95
Memory freed for p21, 40 bytes: 0x5571948c9b95
Memory freed for p22, 42 bytes: 0x5571948c9d83

The first fit malloc for p23: 0x5571948c9440
Memory freed, 10 bytes: 0x5571948c9440

The first fit malloc for p24: 0x5571948c9440
Memory freed, 10 bytes: 0x5571948c9440

The first fit malloc for p25: 0x5571948c946a
Memory freed, 15 bytes: 0x5571948c946a

The first fit malloc for p26: 0x5571948c946a
Memory freed, 12 bytes: 0x5571948c946a

Total amout of memory used: 2030 bytes
```

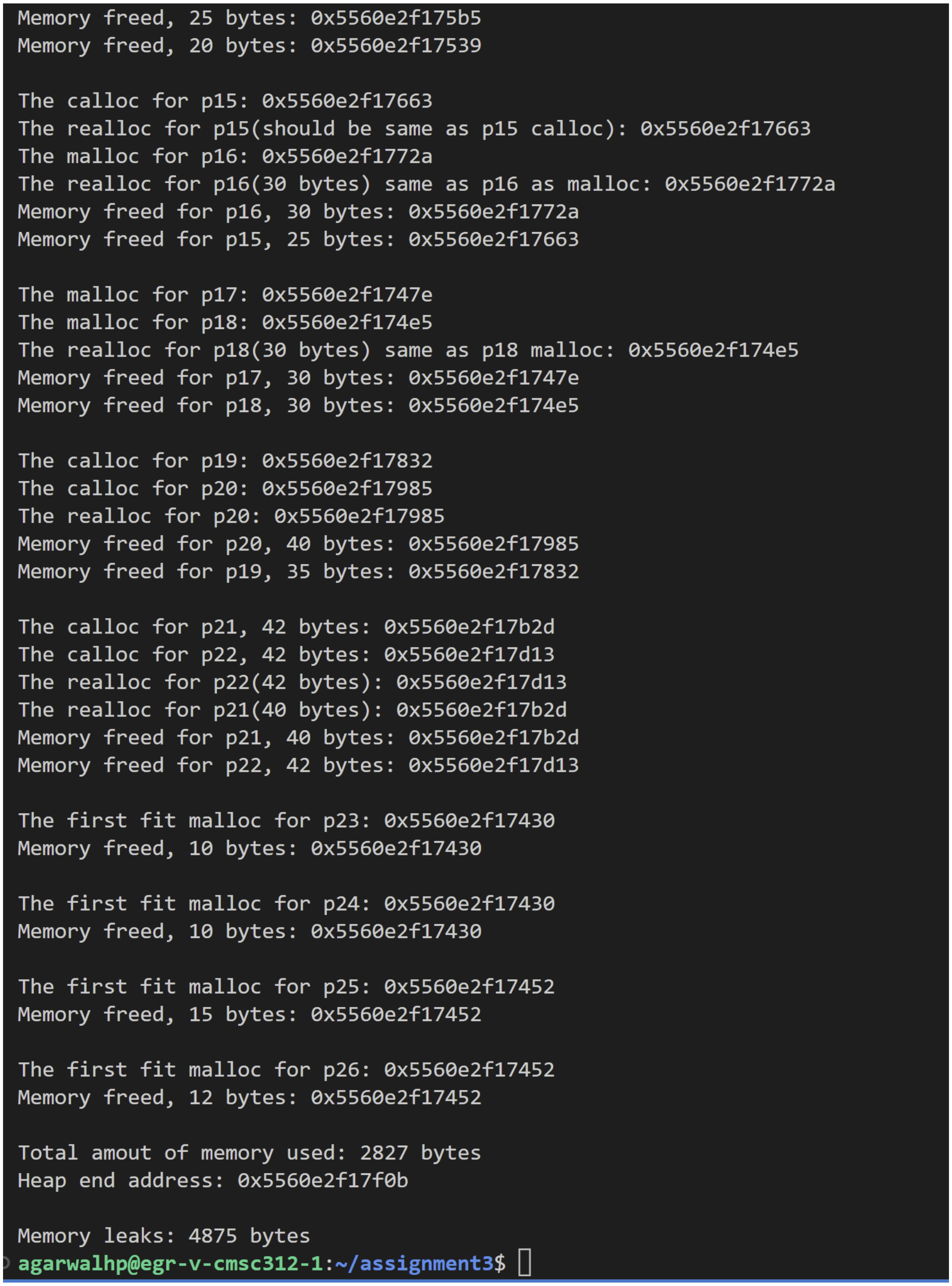


```
Total amout of memory used: 2939 bytes
Heap end address: 0x5571948c9f7b

Memory leaks: 4987 bytes
```

This is the output of ./myprogramOG which is the original implementation of a single linked list.



```
agarwalhp@egr-v-cmsc312-1:~/assignment3$ ./myprogramOG
Heap start address: 0x5560e2f17000

The malloc for p1: 0x5560e2f17430
Memory freed, 10 bytes: 0x5560e2f17430

The malloc for p2: 0x5560e2f17430
Memory freed, 10 bytes: 0x5560e2f17430

The calloc for p3: 0x5560e2f17452
The realloc for p3: 0x5560e2f17452
Memory freed, 20 bytes: 0x5560e2f17452
Memory freed, realloc 20 bytes: 0x5560e2f17452

The malloc for p4: 0x5560e2f17452
Memory freed, 15 bytes: 0x5560e2f17452

The malloc for p5: 0x5560e2f17452
Memory freed, 12 bytes: 0x5560e2f17452

The malloc for p6: 0x5560e2f17452
The realloc for p6(same as malloc p6): 0x5560e2f17452
Memory freed, 20 bytes: 0x5560e2f17452
Memory freed, realloc 20 bytes: 0x5560e2f17452

The malloc for p7: 0x5560e2f1747e
The malloc for p8: 0x5560e2f174b4
Memory freed, 25 bytes: 0x5560e2f174b4
Memory freed, 30 bytes: 0x5560e2f1747e

The calloc for p9: 0x5560e2f1747e
Memory freed, 15 bytes: 0x5560e2f1747e

The best fit for malloc for p10(25 bytes) expected to be same as p8: 0x5560e2f1747e
p10 after realloc should be 20 bytes(same as p10): 0x5560e2f1747e
p10 after realloc should be 15 bytes(same as p10): 0x5560e2f1747e
Memory freed, 15 bytes: 0x5560e2f1747e

The calloc for p11: 0x5560e2f1747e
The calloc for p12: 0x5560e2f174e5
Memory freed, 10 bytes: 0x5560e2f174e5

Realloc should be 20 bytes(same as p11): 0x5560e2f1747e
Memory freed, 10 bytes: 0x5560e2f1747e

The calloc for p13: 0x5560e2f17539
The calloc for p14: 0x5560e2f175b5
```



```
Memory freed, 25 bytes: 0x5560e2f175b5
Memory freed, 20 bytes: 0x5560e2f17539

The calloc for p15: 0x5560e2f17663
The realloc for p15(should be same as p15 calloc): 0x5560e2f17663
The malloc for p16: 0x5560e2f1772a
The realloc for p16(30 bytes) same as p16 as malloc: 0x5560e2f1772a
Memory freed for p16, 30 bytes: 0x5560e2f1772a
Memory freed for p15, 25 bytes: 0x5560e2f17663

The malloc for p17: 0x5560e2f1747e
The malloc for p18: 0x5560e2f174e5
The realloc for p18(30 bytes) same as p18 malloc: 0x5560e2f174e5
Memory freed for p17, 30 bytes: 0x5560e2f1747e
Memory freed for p18, 30 bytes: 0x5560e2f174e5

The calloc for p19: 0x5560e2f17832
The calloc for p20: 0x5560e2f17985
The realloc for p20: 0x5560e2f17985
Memory freed for p20, 40 bytes: 0x5560e2f17985
Memory freed for p19, 35 bytes: 0x5560e2f17832

The calloc for p21, 42 bytes: 0x5560e2f17b2d
The calloc for p22, 42 bytes: 0x5560e2f17d13
The realloc for p22(42 bytes): 0x5560e2f17d13
The realloc for p21(40 bytes): 0x5560e2f17b2d
Memory freed for p21, 40 bytes: 0x5560e2f17b2d
Memory freed for p22, 42 bytes: 0x5560e2f17d13

The first fit malloc for p23: 0x5560e2f17430
Memory freed, 10 bytes: 0x5560e2f17430

The first fit malloc for p24: 0x5560e2f17430
Memory freed, 10 bytes: 0x5560e2f17430

The first fit malloc for p25: 0x5560e2f17452
Memory freed, 15 bytes: 0x5560e2f17452

The first fit malloc for p26: 0x5560e2f17452
Memory freed, 12 bytes: 0x5560e2f17452

Total amout of memory used: 2827 bytes
Heap end address: 0x5560e2f17f0b

Memory leaks: 4875 bytes
agarwalhp@egr-v-cmsc312-1:~/assignment3$
```