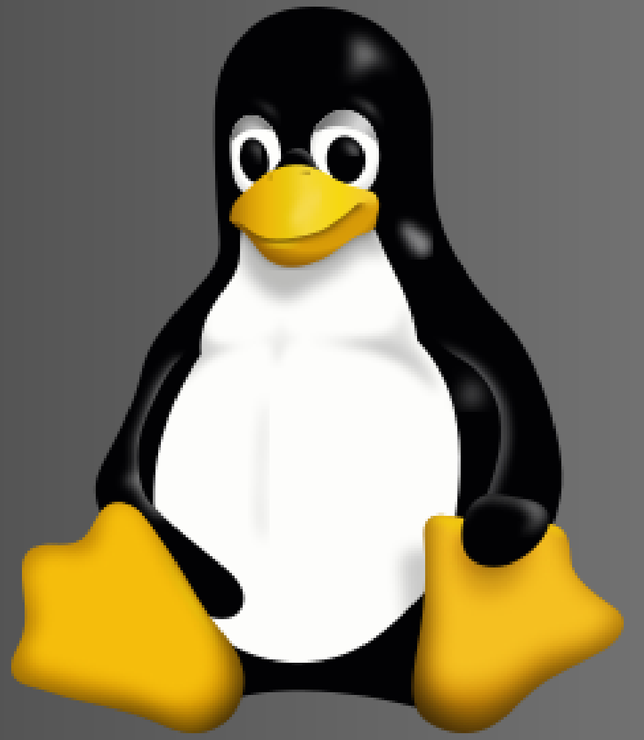
Heap Memory Manager



Project Developed at STMicroelectronics

under Supervision of Eng.Reda Maher

Presented by: Gehad Hassanein

Overview

01
Introduction

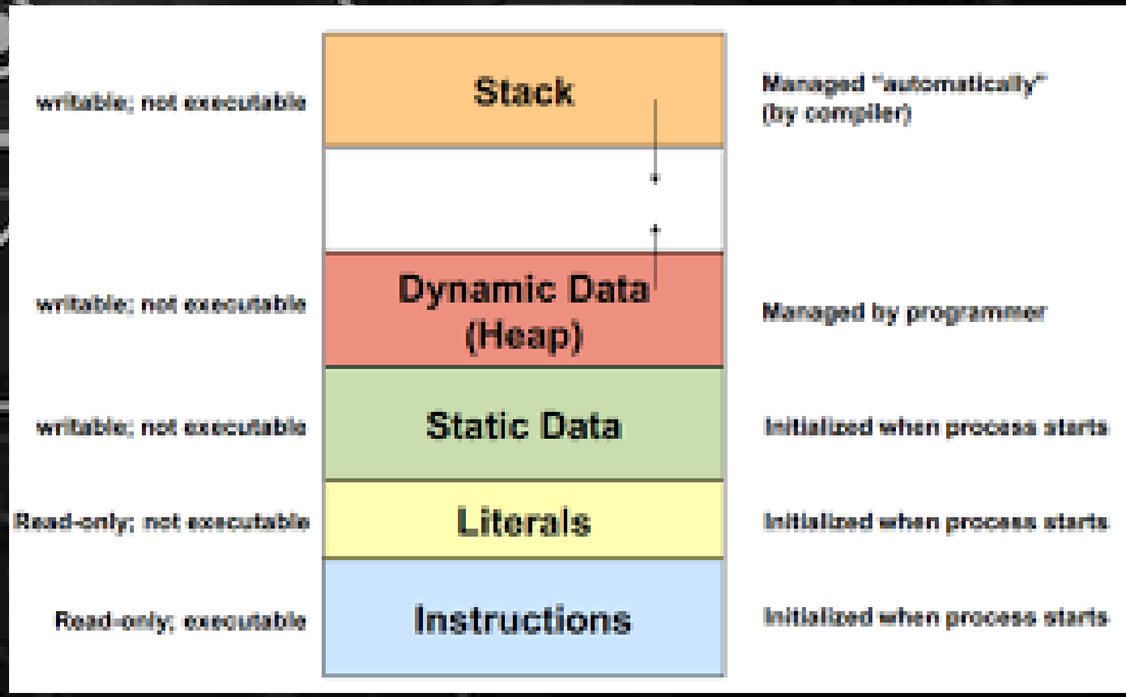
02
Implementation

03
Testing

01

Introduction

- Heap management is an essential aspect of programming in the C language, particularly in Linux environments
- The heap is a region of memory used for dynamic memory allocation, allowing programs to request and release memory at runtime.
- Key goals include dynamic memory allocation, deallocation, implementing various memory management policies, optimizing performance, and integrating error handling mechanisms.
- This project offers essential heap functionalities such as malloc, calloc, realloc, and free



02
Implementation

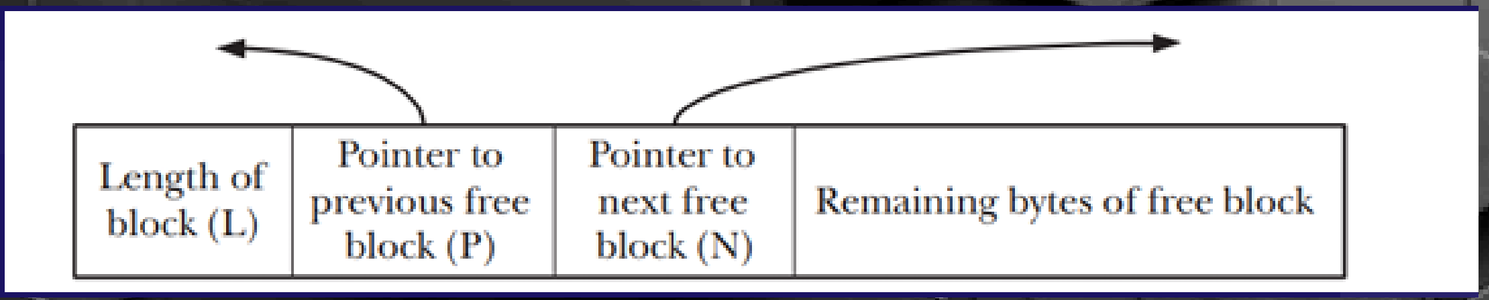
Project Structure

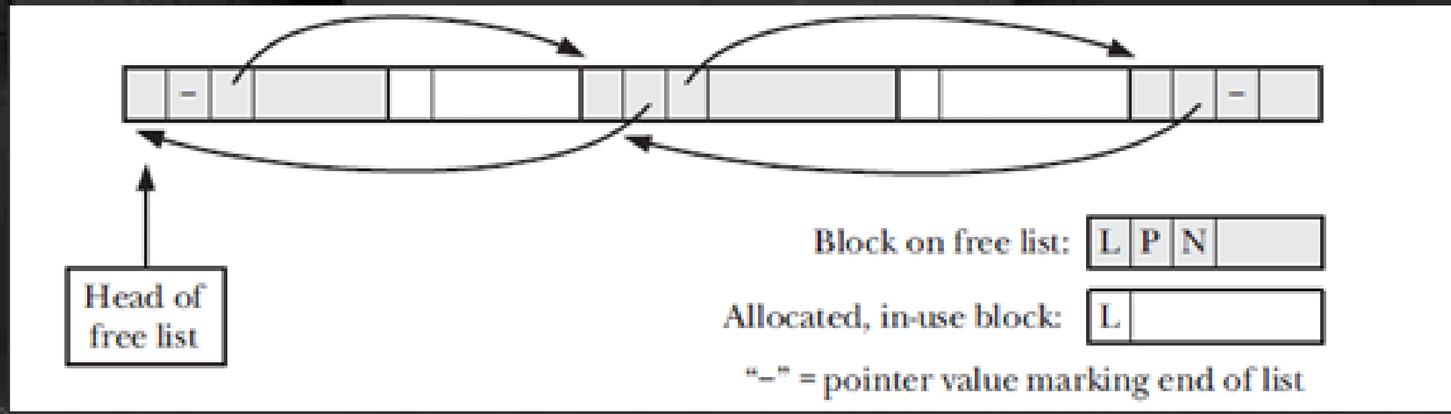
Freed block

```
typedef struct  {
    uint32_t size;
    Block_t *head;
} FreeList_t;
```

```
typedef struct block {
    uint32_t length;
    struct block *previous;
    struct block *next;
} Block_t;
```

Allocation and Deallocation





Main Interfaces

Malloc

malloc is a function in C that dynamically allocates memory during runtime

1- Search for a Free Block:

- malloc searches for a suitable free memory block in the free list using strategies like First-fit or Best-fit.

2- Allocation:

- If a suitable block is found, it is marked as used, and a pointer to the usable memory is returned.

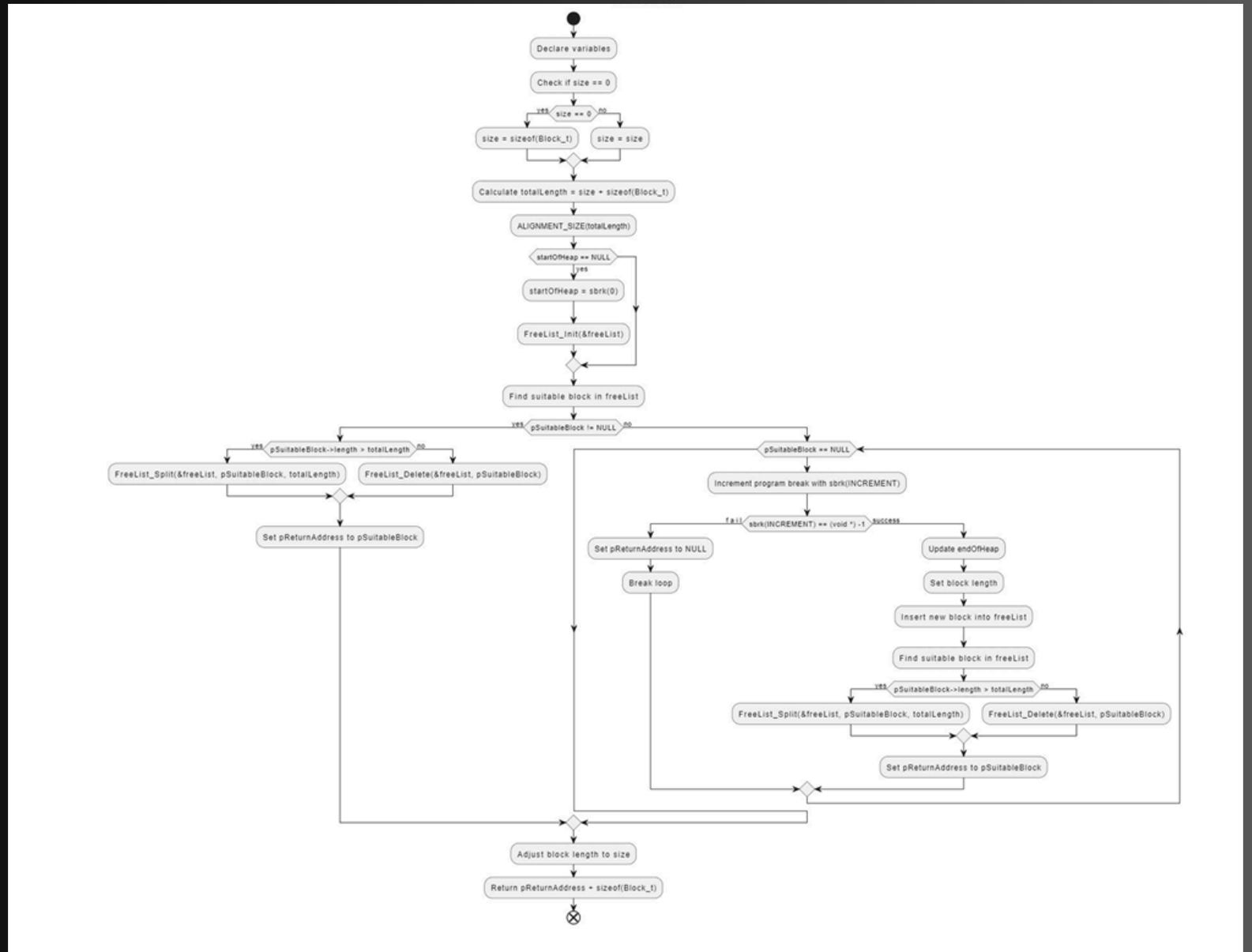
3- Heap Expansion:

- If no suitable block is available, malloc extends the available memory:
  - Using sbrk(): Increases the program break to grow the heap.
  - Or Using mmap(): Directly allocates memory from the OS, useful for larger allocations or to avoid heap fragmentation.

4- Memory Management:

- malloc keeps track of allocated and free blocks, managing memory efficiently to minimize fragmentation and ensure performance.

malloc () Flow Chart



Free

free is a function in C is used to deallocate memory previously allocated with malloc or calloc

1- Validate the Pointer:

- realloc first checks if the given pointer is valid and points to a previously allocated block.

2- Insert into Free List:

- The block is inserted back into the free list using FreeList_Insert(), making it available for future allocations.

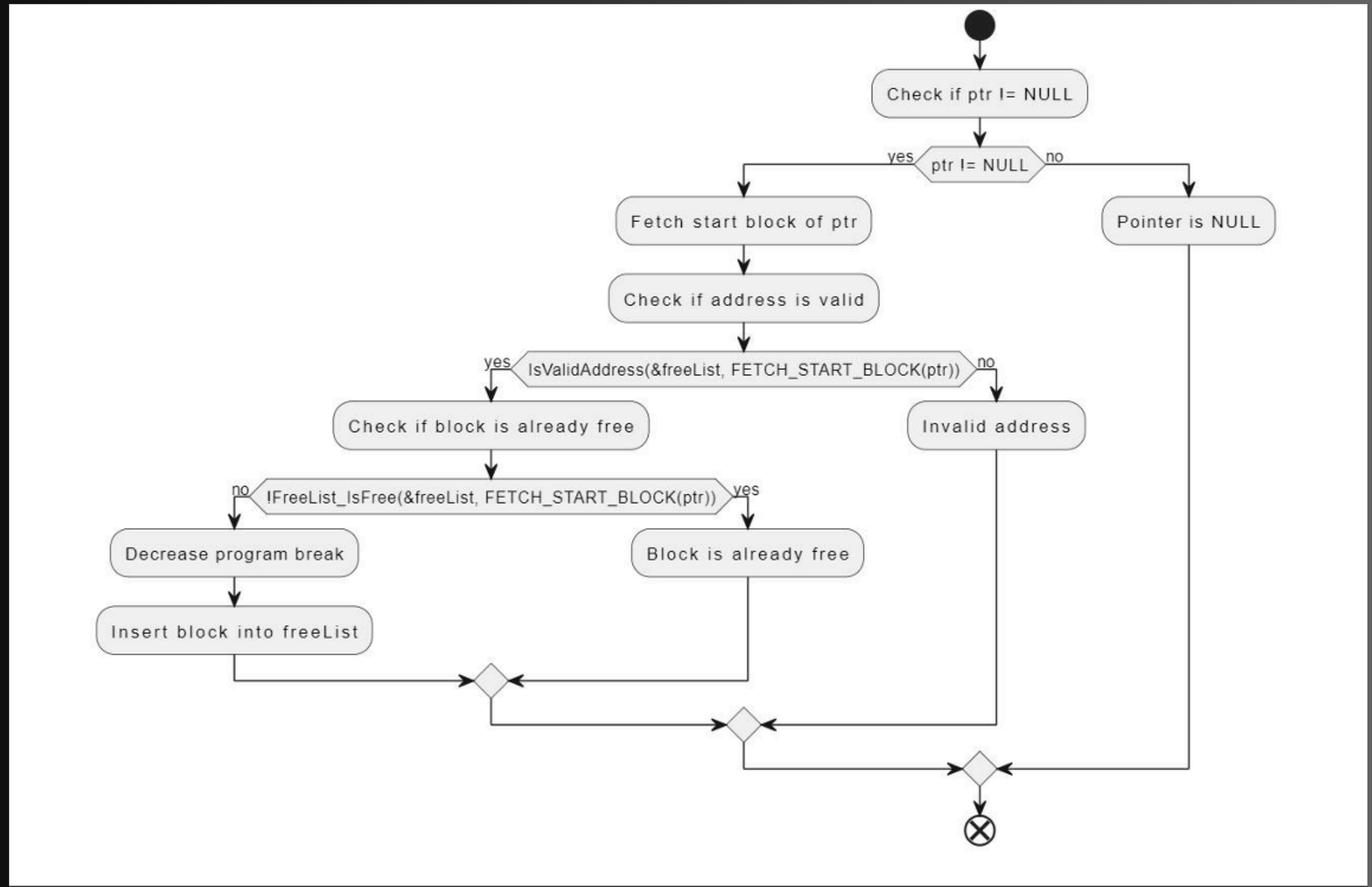
3- Merge Adjacent Free Blocks:

- Adjacent free blocks are merged to reduce fragmentation and create larger contiguous memory blocks, optimizing memory usage.

4- Release Unused Memory:

- The program break is lowered when possible, releasing unused memory back to the OS to reduce memory footprint, prevent leaks, and optimize address space.

free () Flow Chart



Realloc

realloc is a function in C used to dynamically resize a previously allocated block of memory

1- Validate the Pointer:

- realloc first checks if the given pointer is valid and points to a previously allocated block.

2- Determine New Size:

- If the new size is zero, realloc behaves like free, releasing the memory block.
- If the pointer is NULL, realloc behaves like malloc, allocating a new block of the specified size.

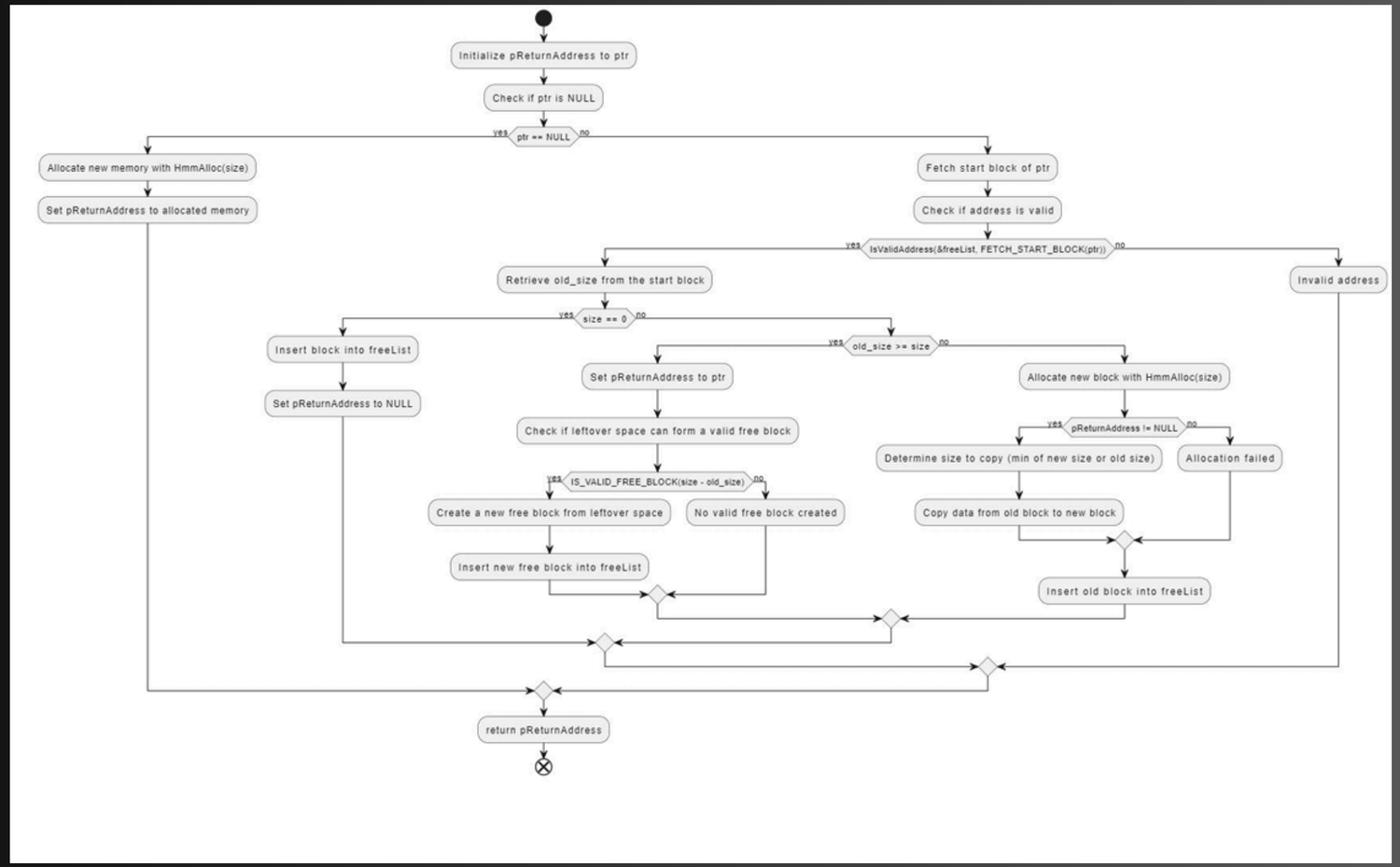
3- Resize Memory Block:

- If the existing block has enough space to accommodate the new size, it is resized in place, preserving the existing data.
- If the block cannot be resized in place, a new block is allocated:
  - Data from the old block is copied to the new block.
  - The old block is freed, and the pointer is updated to the new location.

4- Return Updated Pointer:

- Returns a pointer to the resized memory block, or NULL if the allocation fails, without altering the original data.

realloc () Flow Chart



Calloc

calloc is a function in C used to dynamically allocate and initialize memory by zero

1- Calculate Total Memory Size:

- calloc takes two parameters: the number of elements and the size of each element.
- It calculates the total memory required by multiplying these two values.

2- Allocate Memory:

- It requests a block of memory of the calculated size using a method similar to malloc.

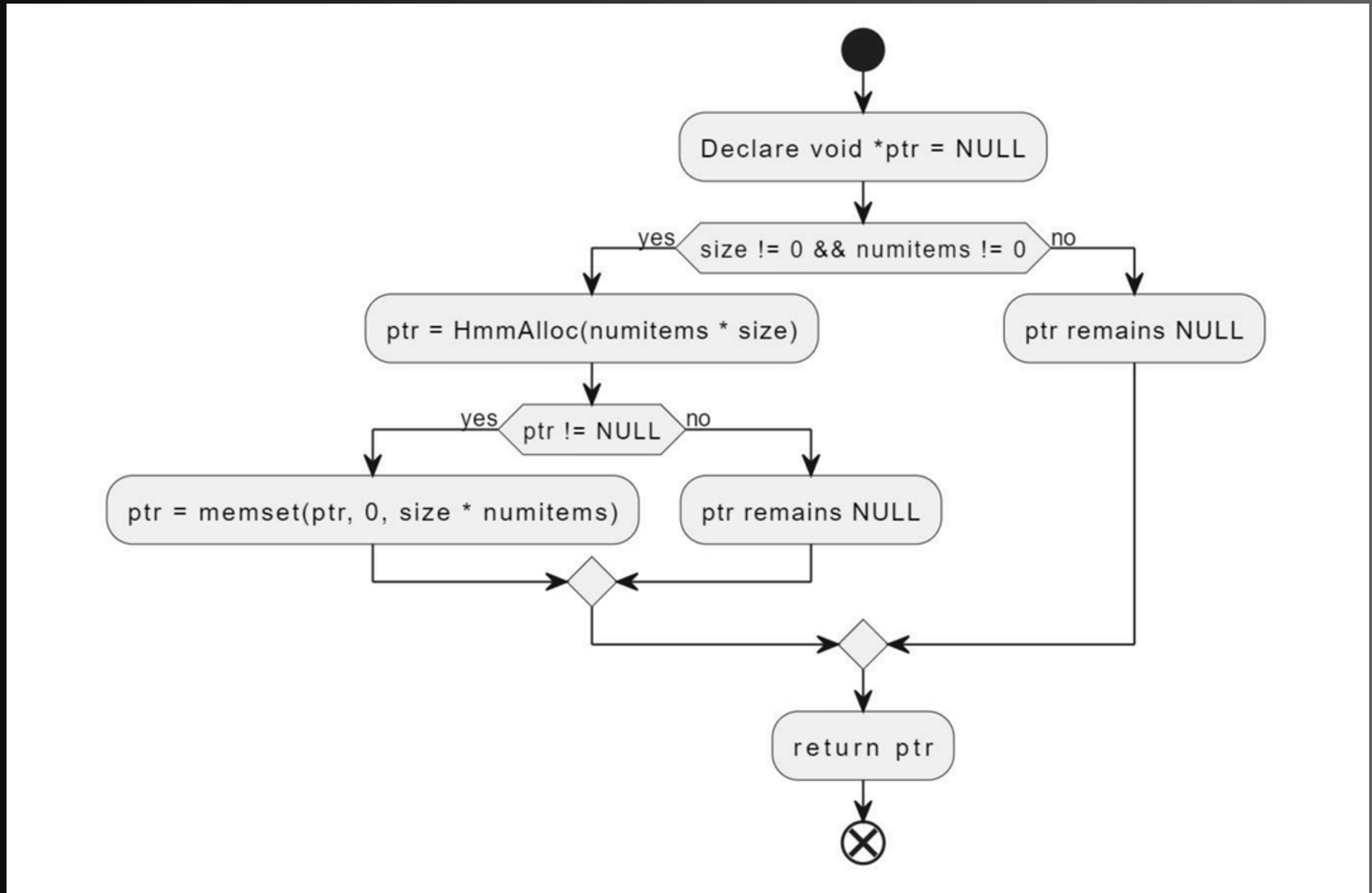
3- Initialize Memory to Zero:

- Unlike malloc, calloc initializes all allocated memory bytes to zero, ensuring that the allocated memory is clean.

4- Return Pointer:

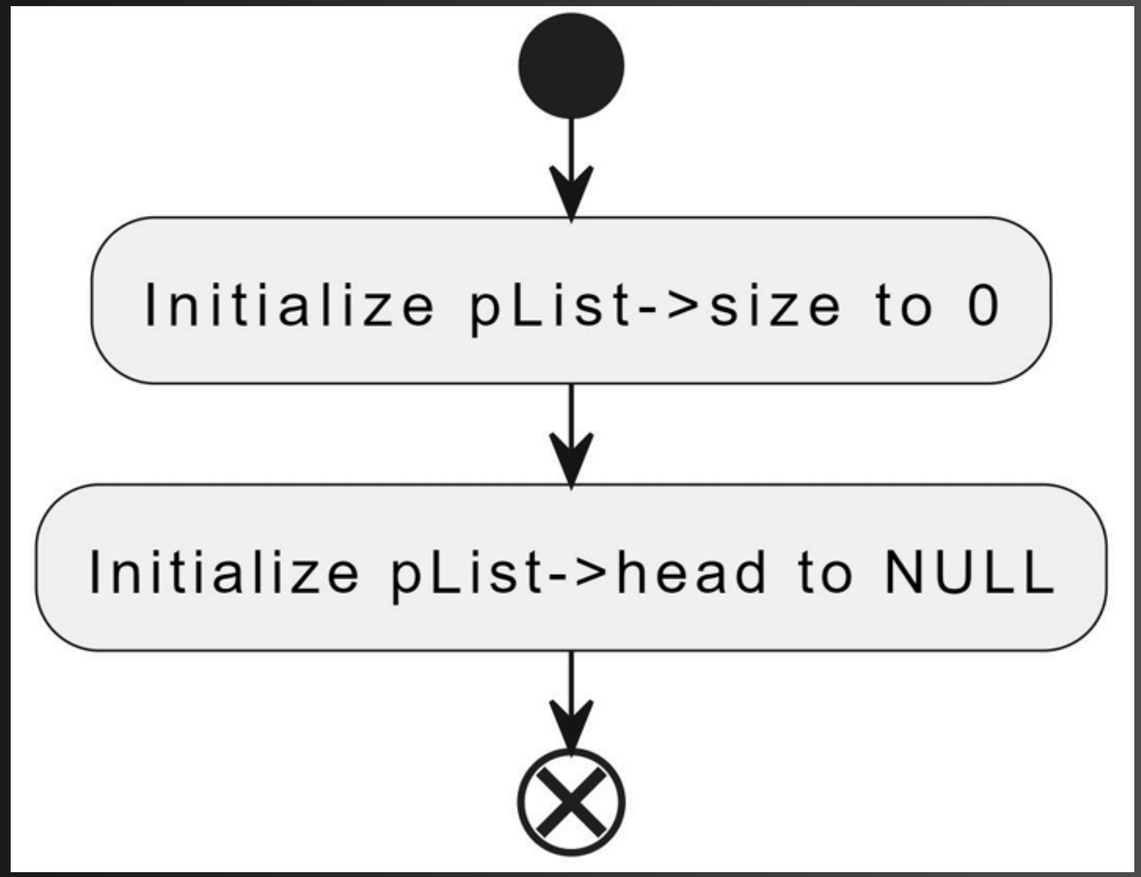
- Returns a pointer to the beginning of the allocated memory block or NULL if the allocation fails.

calloc () Flow Chart

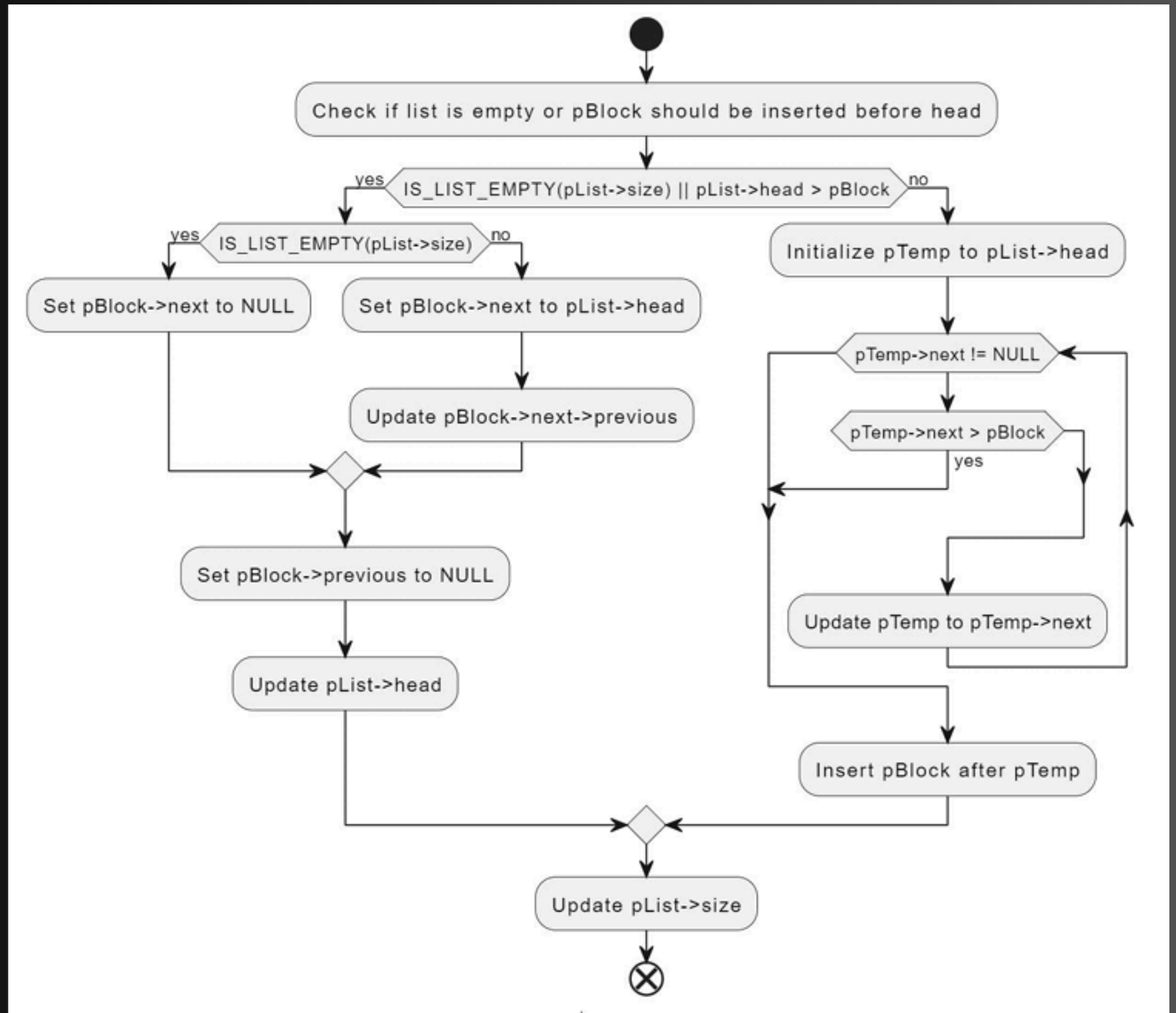


FreeList Interfaces

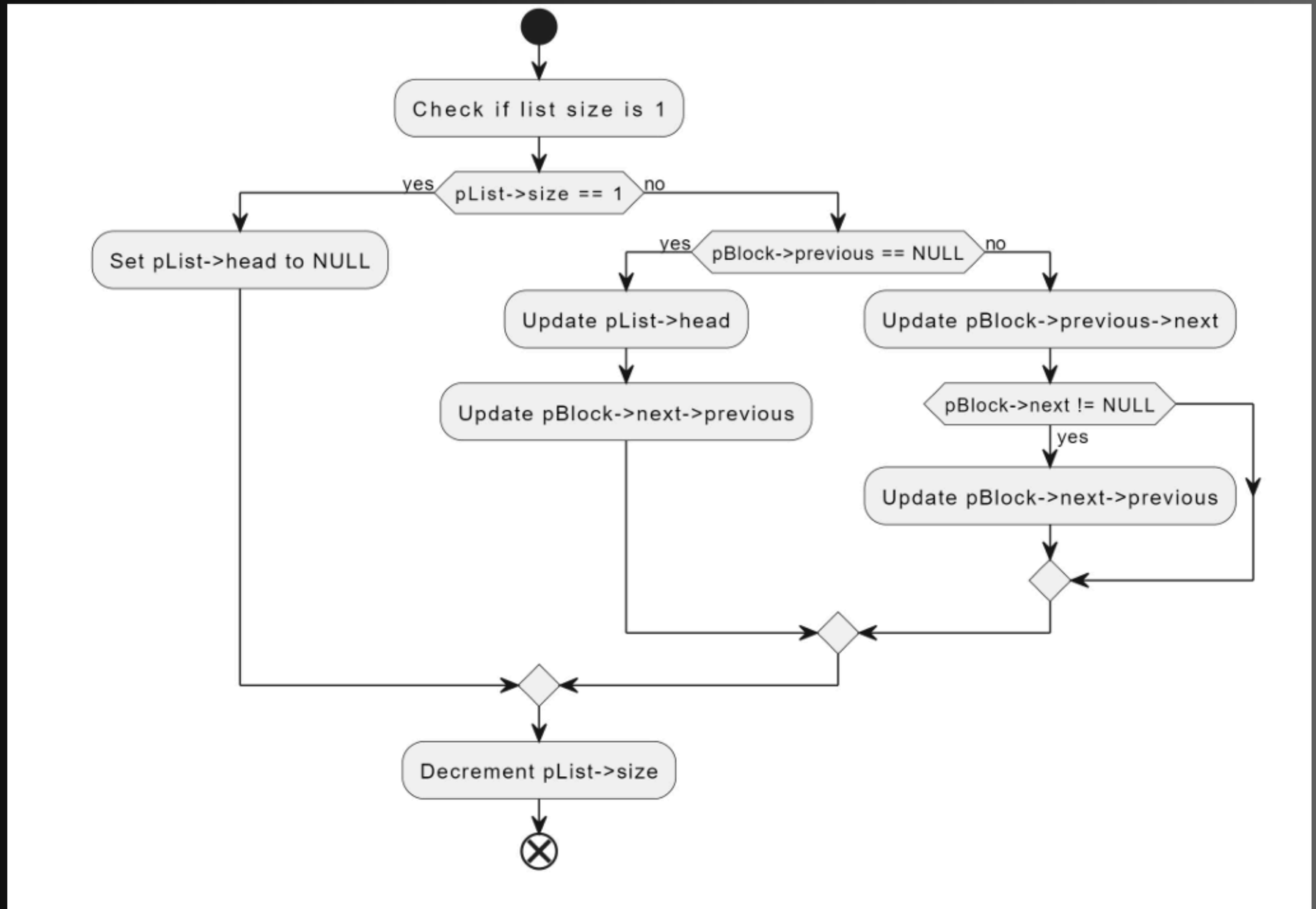
FreeList_Init () Flow Chart



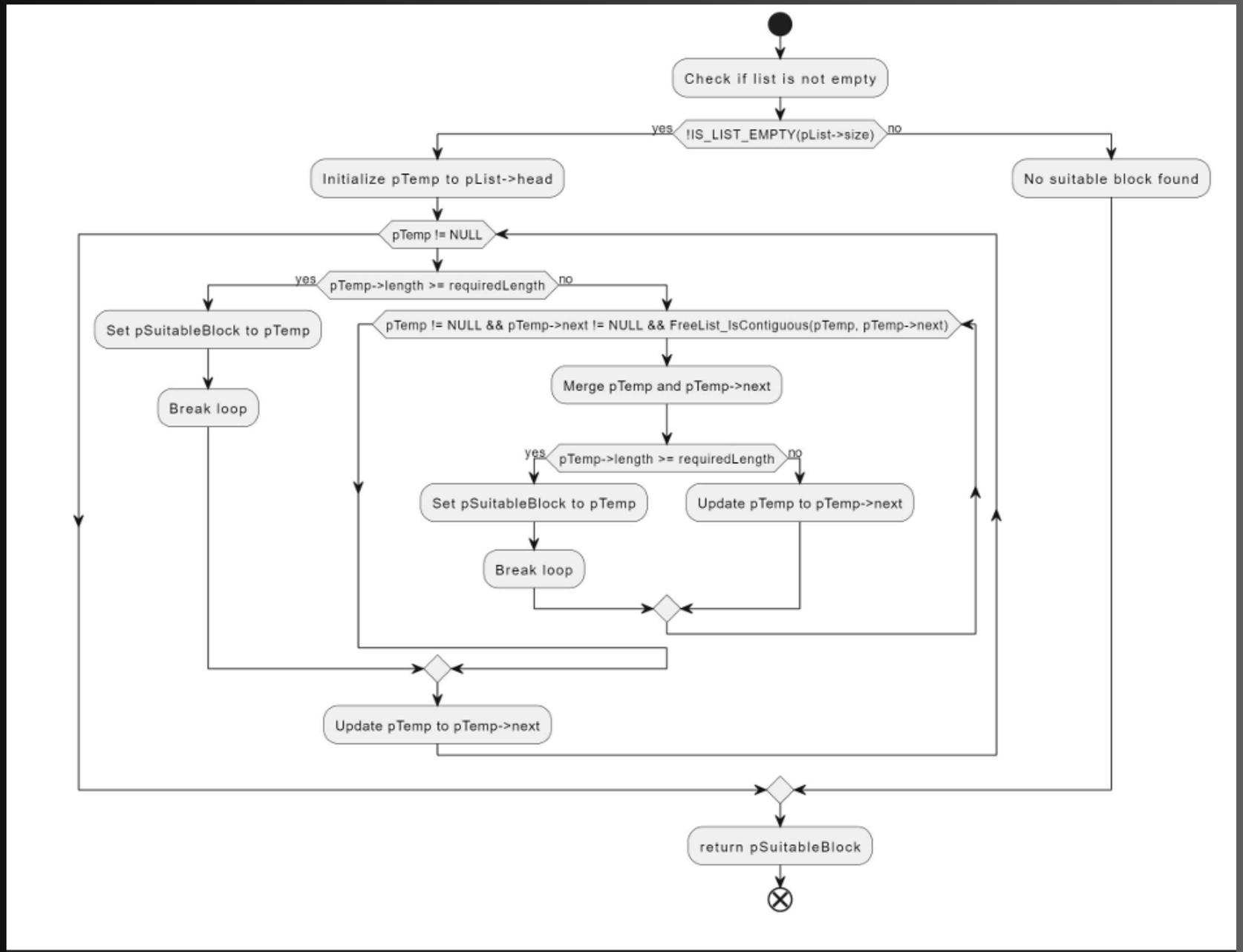
FreeList_Insert () Flow Chart



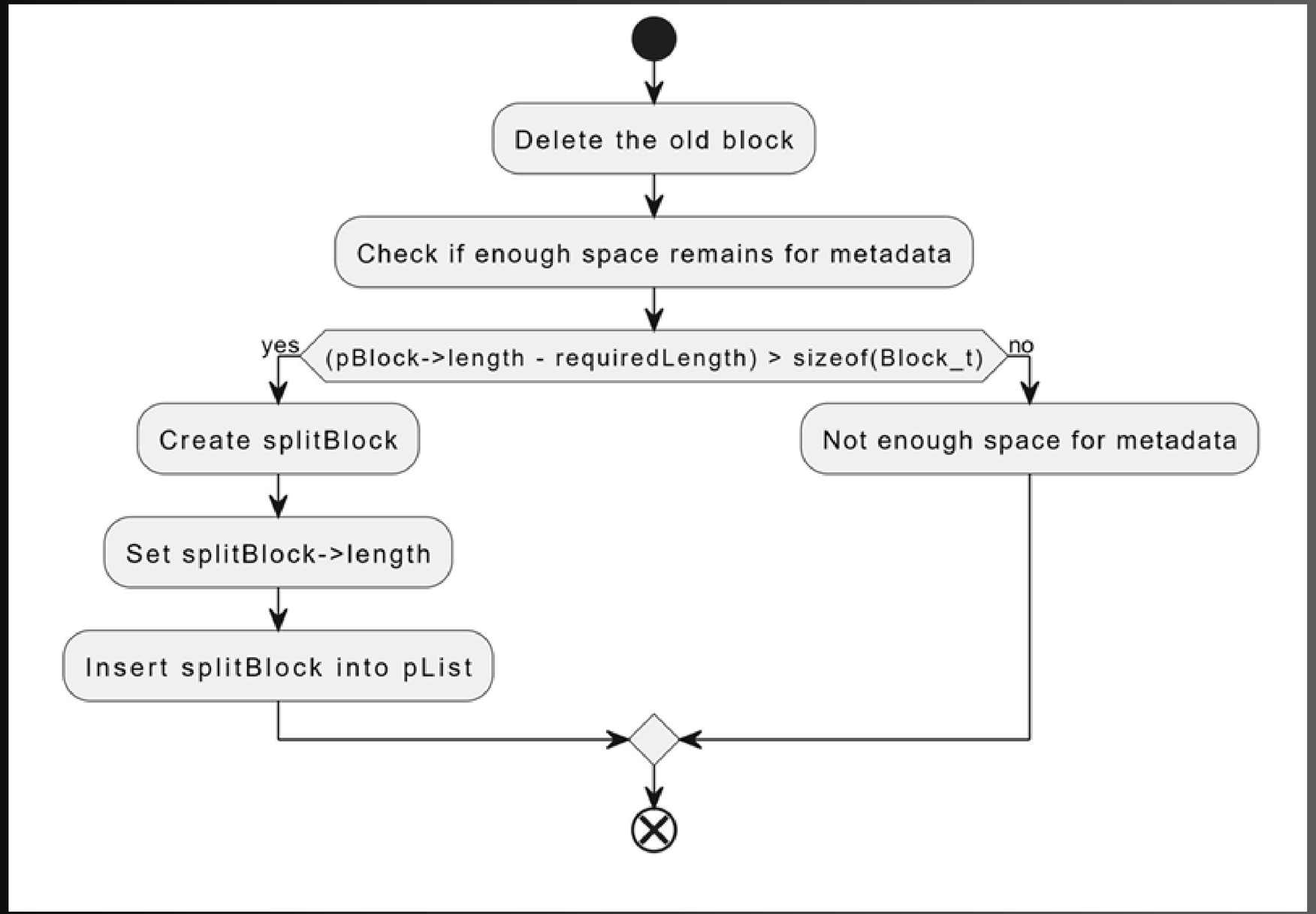
FreeList_Delete () Flow Chart



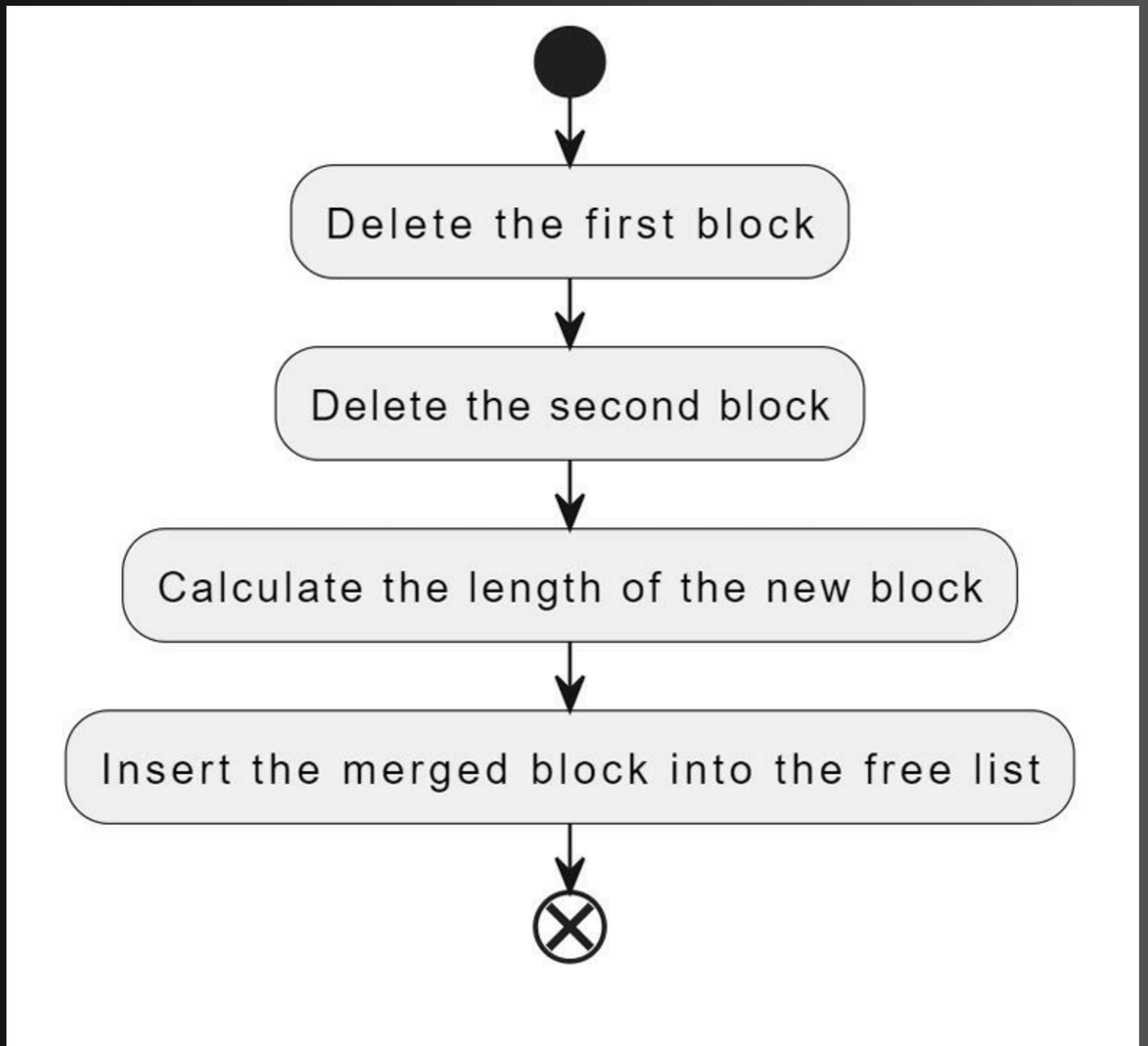
FreeList_FindSuitableBlock () Flow Chart



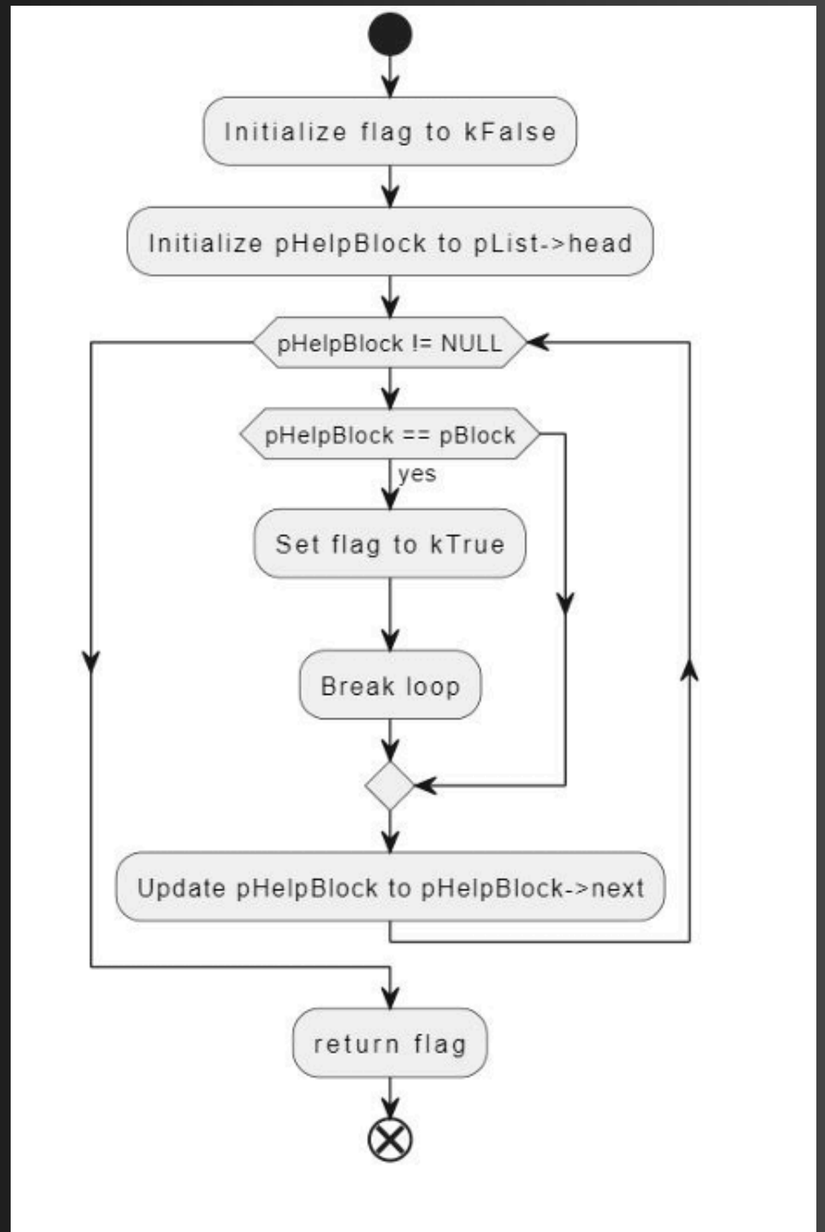
FreeList_Split () Flow Chart



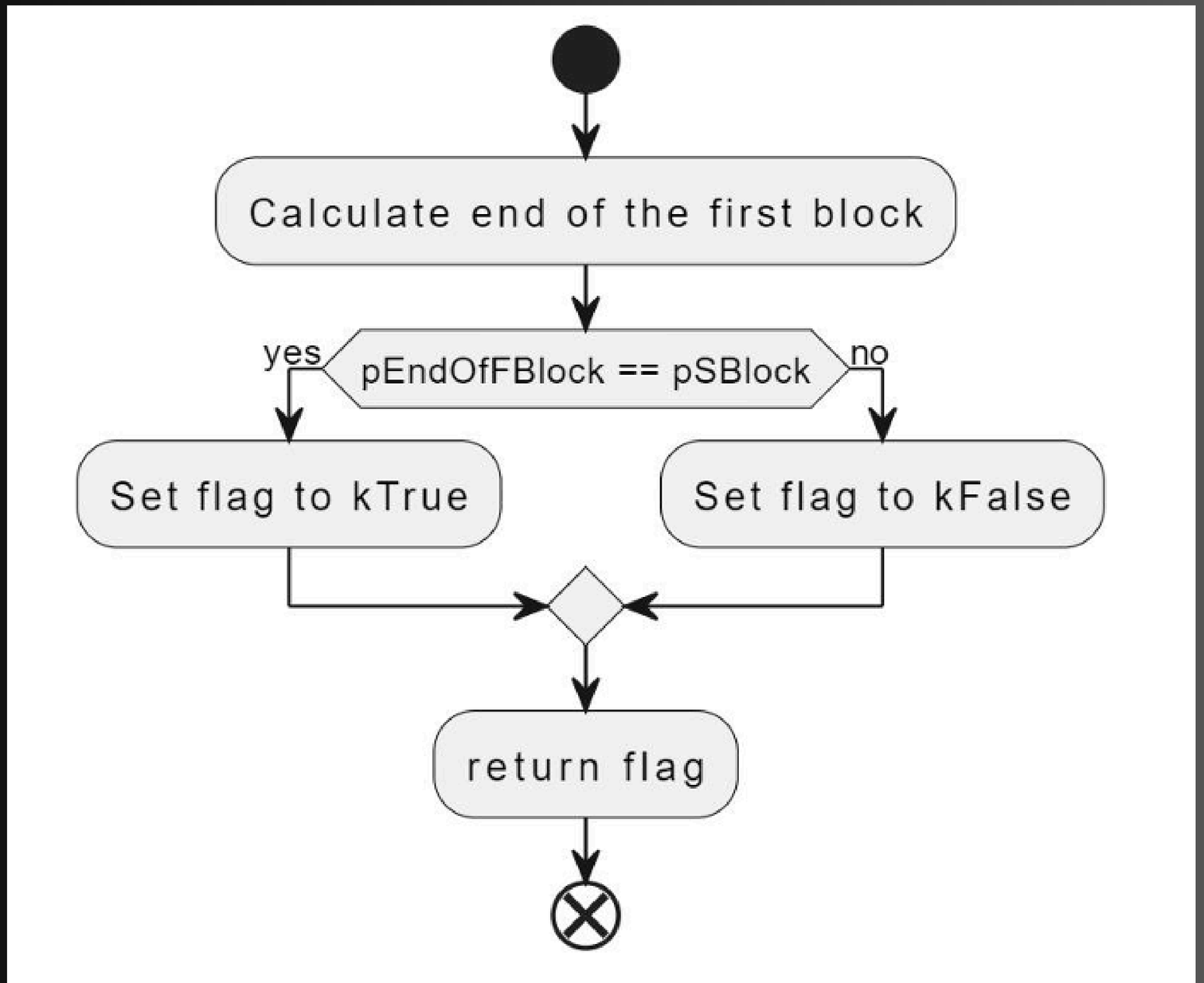
FreeList_Merge () Flow Chart



FreeList_IsFree () Flow Chart

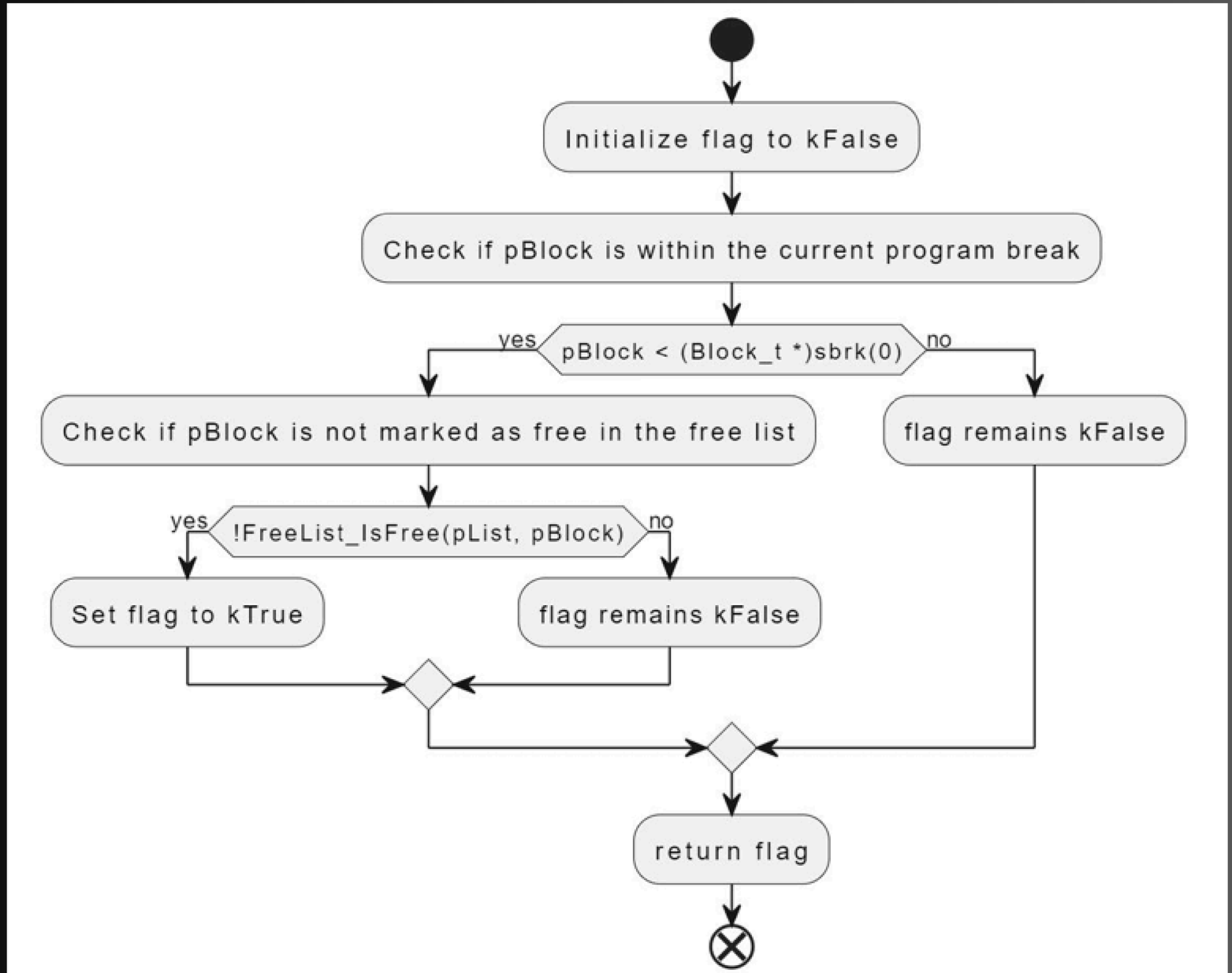


FreeList_IsContiguous () Flow Chart

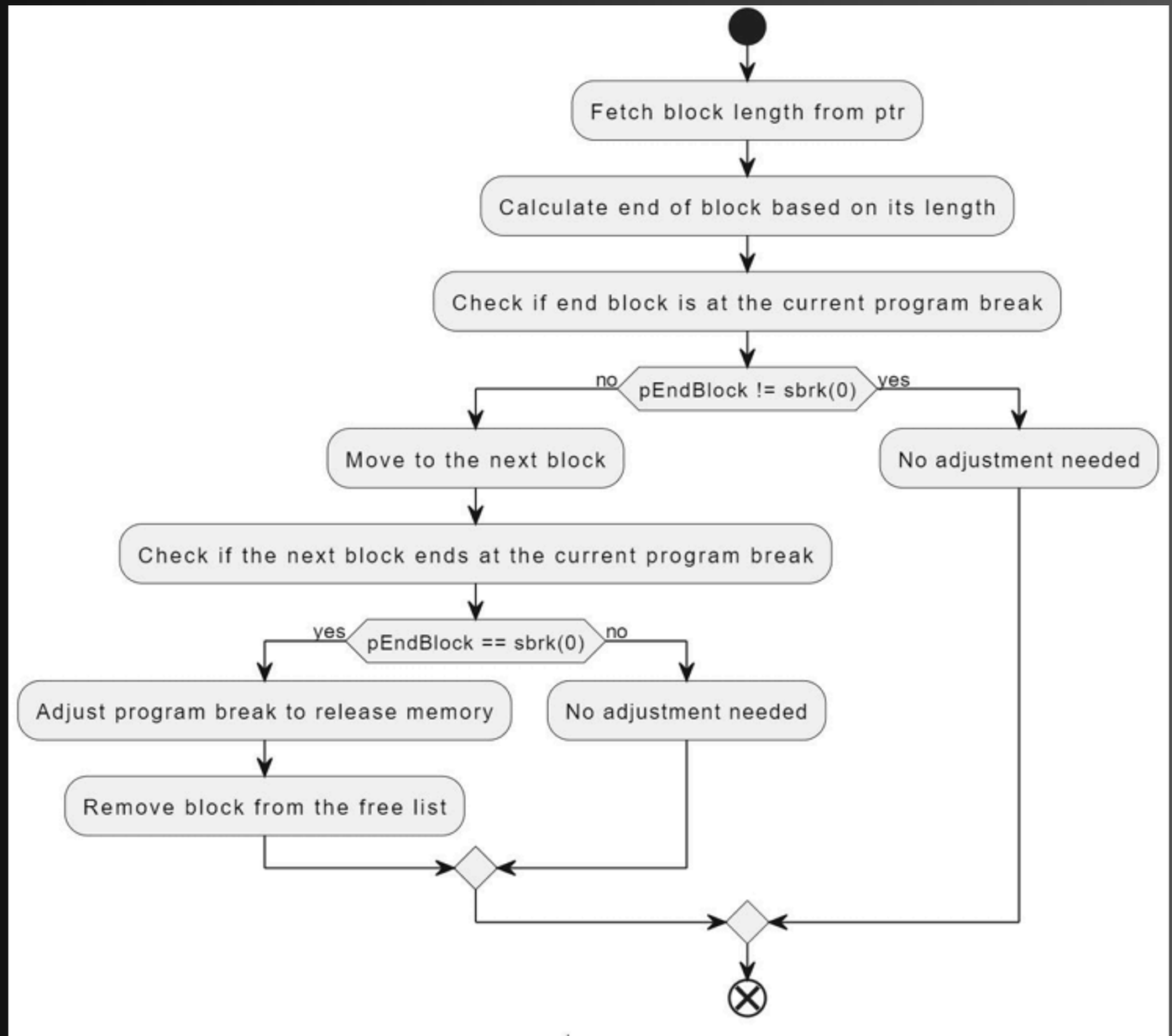


Helper Interfaces

IsVaildAddress () Flow Chart



DecreaseProgramBreak () Flow Chart



03
Testing

Randomize test

```
Freeing memory at address 0x60a9d3bf86b8
Allocated memory of size 4878 at address 0x60a9d4063f28
Allocated memory of size 2081 at address 0x60a9d378ad68
Freeing memory at address 0x60a9d42c8ef0
Allocated memory of size 10220 at address 0x60a9d5714360
Allocated memory of size 7313 at address 0x60a9d4beaf30
Allocated memory of size 4091 at address 0x60a9d3e0e0e8
Freeing memory at address 0x60a9d519dea0
Allocated memory of size 4706 at address 0x60a9d42b2f08
Freeing memory at address 0x60a9d45a2d98
Allocated memory of size 2303 at address 0x60a9d3a0f278
Freeing memory at address 0x60a9d36efae8
Allocated memory of size 3800 at address 0x60a9d3bf86b8
Freeing memory at address 0x60a9d526da20
Allocated memory of size 9071 at address 0x60a9d519dea0
Allocated memory of size 9543 at address 0x60a9d5716b68
Allocated memory of size 6143 at address 0x60a9d489b998
Freeing memory at address 0x60a9d38a4ca8
Allocated memory of size 9843 at address 0x60a9d57190c8
Freeing memory at address 0x60a9d47b36b0
Freeing memory at address 0x60a9d3c277c8
Freeing memory at address 0x60a9d41524e8
Allocated memory of size 2036 at address 0x60a9d36efae8
Freeing memory at address 0x60a9d4575fa0
Allocated memory of size 10207 at address 0x60a9d571b758
Freeing memory at address 0x60a9d39a4f78
Allocated memory of size 2700 at address 0x60a9d39a4f78
Freeing memory at address 0x60a9d41774d8
Freeing memory at address 0x60a9d3782960
Allocated memory of size 9578 at address 0x60a9d571df50
Freeing memory at address 0x60a9d3c3a088
Freeing memory at address 0x60a9d406eb90
Allocated memory of size 5740 at address 0x60a9d406eb90
Freeing memory at address 0x60a9d3dc8fe8
Allocated memory of size 2372 at address 0x60a9d3a76790
Freeing memory at address 0x60a9d56cbcd8
Allocated memory of size 3514 at address 0x60a9d3c3a088
Allocated memory of size 4765 at address 0x60a9d41524e8
Allocated memory of size 1957 at address 0x60a9d37dd8a0
Allocated memory of size 7210 at address 0x60a9d47b36b0
Allocated memory of size 2432 at address 0x60a9d3aabda8
Allocated memory of size 8846 at address 0x60a9d51bbbe0
Allocated memory of size 8205 at address 0x60a9d51a0400
Allocated memory of size 2385 at address 0x60a9d3a78308
Freeing memory at address 0x60a9d4a92db8
Freeing memory at address 0x60a9d3603928
Freeing memory at address 0x60a9d37b4ad8
Freeing memory at address 0x60a9d4dafdc8
```

```
Freeing remaining memory at address 0x555556864a62
Freeing remaining memory at address 0x555555932058
Freeing remaining memory at address 0x555556de6d82
Freeing remaining memory at address 0x555555897adf
Freeing remaining memory at address 0x55555560b100
Freeing remaining memory at address 0x555555d558d9
Freeing remaining memory at address 0x555555639ab0
Freeing remaining memory at address 0x5555561518a3
Freeing remaining memory at address 0x5555560b5b10
Freeing remaining memory at address 0x555555adc5db
Freeing remaining memory at address 0x555555cb0941
Freeing remaining memory at address 0x5555558b1a58
Freeing remaining memory at address 0x555555d19a4e
Freeing remaining memory at address 0x555555fb6911
Freeing remaining memory at address 0x555555f9c3c3
Freeing remaining memory at address 0x55555597e844
Freeing remaining memory at address 0x555555d26e66
Freeing remaining memory at address 0x5555556f1f1b
Freeing remaining memory at address 0x555557996076
Freeing remaining memory at address 0x555556b43fec
Freeing remaining memory at address 0x55555623045d
Freeing remaining memory at address 0x555555fff650
Freeing remaining memory at address 0x5555562b803f
Freeing remaining memory at address 0x5555557e8238
Freeing remaining memory at address 0x555556421a33
Freeing remaining memory at address 0x5555565c8f17
Freeing remaining memory at address 0x555556a2556f
Freeing remaining memory at address 0x55555758d4e4
Freeing remaining memory at address 0x55555564c87d
Freeing remaining memory at address 0x5555558a2b24
Freeing remaining memory at address 0x555556ef351a
Freeing remaining memory at address 0x555555869a0f
Freeing remaining memory at address 0x555556eb6058
Freeing remaining memory at address 0x555557855007
Freeing remaining memory at address 0x5555557f271c
Freeing remaining memory at address 0x55555599e8a7
Freeing remaining memory at address 0x555555700bee
Freeing remaining memory at address 0x555555b27472
Freeing remaining memory at address 0x5555556d04e4
Freeing remaining memory at address 0x555556ed0260
Freeing remaining memory at address 0x555556f49335
Freeing remaining memory at address 0x5555557db747
Freeing remaining memory at address 0x5555558283be
Test complete.
```

Linux utilities test

```
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ make run-shared-lib COMMAND=ls
FlowChart.pdf  inc  libhmm.so  main.c  Makefile  README.md  src  test.c
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ make run-shared-lib COMMAND=ls | grep i
inc
libhmm.so
main.c
Makefile
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ make run-shared-lib COMMAND=pwd
/home/gehad/Desktop/Heap-Memory-Manager
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ make run-shared-lib COMMAND=bash
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ ls
FlowChart.pdf  inc  libhmm.so  main.c  Makefile  README.md  src  test.c
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ man 2 printf
No manual entry for printf in section 2
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ pwd
/home/gehad/Desktop/Heap-Memory-Manager
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$ exit
exit
gehad@gehad:~/Desktop/Heap-Memory-Manager$
gehad@gehad:~/Desktop/Heap-Memory-Manager$
```


I fear no man.
But that thing...
Segmentation fault
(core dumped)
It scares me.

Thank you

Many thanks to Eng Reda Maher

□□□□