# M1522.000800 System Programming, Fall 2017

# Malloc Lab: Writing a Dynamic Storage Allocator

## Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 1.Implementation

Before the implementation of the routine, I make some macro definition to make the coding easy to read and write.

macro definition:

#define PACK(size, alloc) ((size)|(alloc))

#define WSIZE 4

#define DSIZE 8

… … … …

## Part 1 malloc part:

Firstly I initialize the memory using function mm\_init(void \*ptr); in this function:

1. Create the prologue and epilogue.
2. Extend the heap.

After mm\_init, I can malloc space for users by mm\_malloc(); in this function,

1. We need maintain alignment, so do the data alignment firstly.
2. Find the suitable place to the request block.  find\_fit()
3. Place the block at the suitable place.  place()
4. If not enough space, extend the heap.

find\_fit(): there are several ways to do this part.

1. First fit; … … … …
2. Next fit; … … … …
3. Best fit; … … … …

Place() :

1. update the info of the header and footer.
2. Whether there is the rest block.

… … … …

… … … …

… … … …

## Part 2 free part:

mm\_free(): frees a block pointed by ptr. Should have been returned by an earlier call to mm\_malloc or mm\_realloc:

1. Update the info of header and footer
2. Coalesce.

… … … …

… … … …

… … … …

Coalesce() : there are 4 conditions when coalesces. … … … …

1. No allocated block + block + no allocated block
2. No allocated block + block + allocated block
3. Allocated block + block + allocated block
4. Allocated block + block + no allocated block

… … … …

… … … …

… … … …

## Part 3 realloc part:

mm\_realloc() : changing the size of allocated block.

Assume that mm\_maloc(50):

1. If mm\_realloc(30), 30 < 50, the similar with place();
2. Else mm\_realloc(100), 100 > 50, find another free block to fit these 100 size block, and then copy the original data to the realloc address.

… … … …

… … … …

… … … …

## 2.Difficulties and Thoughts

For me, the most difficult part is improving the performance.

In the find\_fit() function, I just using the first fit algorithm. But there are 3 types: first fit, next fit and best fit. If I can implement the next fit or best fit algorithm, I could get a better score.

And at the beginning, I was confused about p and bp. So I missed at HDRP(PREV\_BLKP(bp)), GET\_ALLOC(HDRP(PREV\_BLKP(bp))). Then I go back to the macro definition, and realized that it is really useful and important.

… … … …

… … … …

… … … …