Lecitation 16

Monday, July 17th, 2017

Grades Released

Quiz 5: Released after lecitation

HW08: Released (probably) late tonight

You will have all grades back, except for HW11, by **next Tuesday**

Homework 09 Demos

Go to the "Sign-up" tab on T-Square and select a time with any TA

If no times work for you, please contact Preston to schedule a better time **before Tuesday (July 18th)**

Make Up Demos

Go to the "Sign-up" tab on T-Square and select a time with any TA

Will be posted on **Friday (July 21st)**, with times next **Monday (24th) and Tuesday (25th)**

If no times work for you, please contact Preston to schedule a better time **before Monday (24th)**

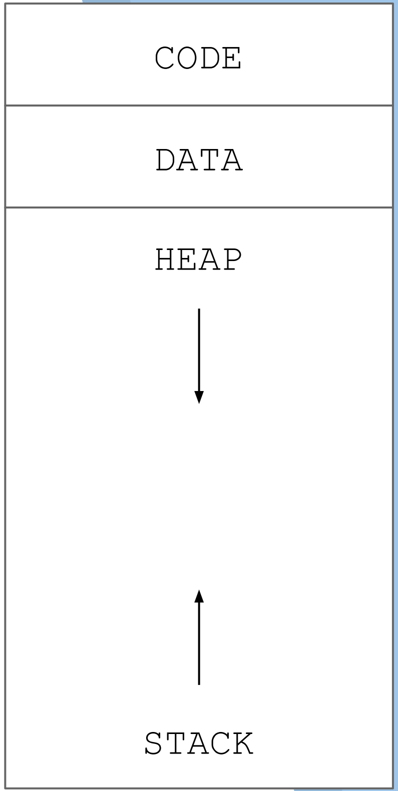
Timed Lab 04

- Dynamic Memory Allocation (`malloc` / `free`)
- **Wednesday, July 19th**, in recitation
- Entire period (1 hr. 45 min.)
- May only reference assignments submitted to T-Square (no Internet, etc.)

Best prep. material will be Homework 10 (!)

Homework 11

- Malloc implementation
- Due on **Tuesday, July 25th @ 11:55pm**
- Posted under "Assignments" on T-Square



Course Instructor Opinion Survey (CIOS)

Please complete by August 6th!

Looking Ahead...

MONDAY (17th)	TUESDAY (18th)	WEDNESDAY (19th)	THURSDAY (20th)	FRIDAY (21th)
HW 09 Demos				
Lecitation 16 ("Stack smashing")	*Lecture*	**Timed Lab 4**	*Lecture*	...
MONDAY (24th)	**TUESDAY (25th)**	**...**	**THURSDAY (3rd)**	
Make Up Demos		...	**Final Exam**	
Lecitation 17 ("Final Exam prep.")	*Lecture* **HW 11 Due**			

Looking Ahead...

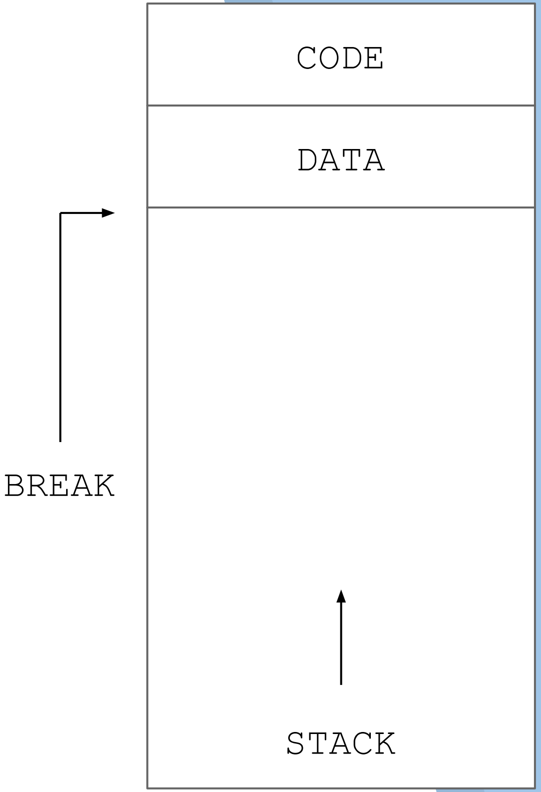
MONDAY (17th)	TUESDAY (18th)	WEDNESDAY (19th)	THURSDAY (20th)	FRIDAY (21th)
HW 09 Demos				
Lecitation 16 ("Stack smashing")	*Lecture*	**Timed Lab 4**	*Lecture*	...
MONDAY (24th)	**TUESDAY (25th)**	**...**	**THURSDAY (3rd)**	
Make Up Demos		...	**Final Exam**	
Lecitation 17 ("Final Exam prep.") 	*Lecture* **HW 11 Due**			

Questions?

Homework 11

Change end of the process's data segment:

- `brk()` - by specifying an address
- `sbrk()` - by specifying a size

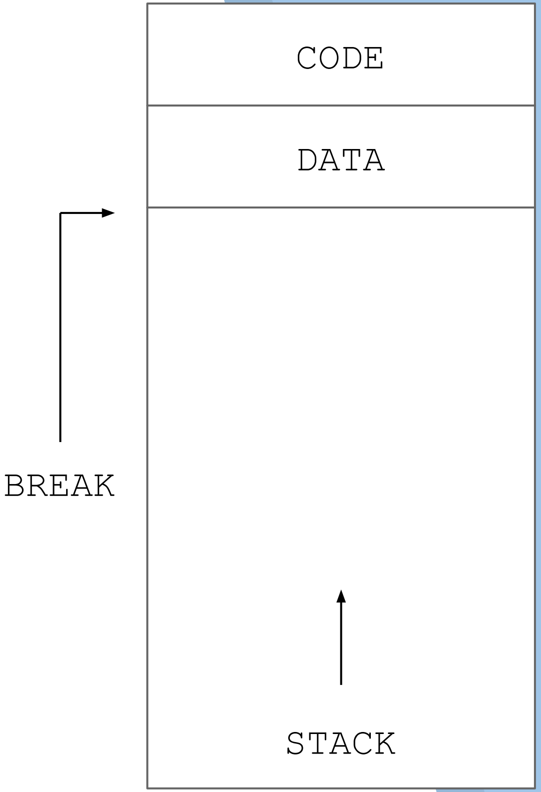


Homework 11

Change end of the process's data segment:

- `brk()` - by specifying an address
- `sbrk()` - by specifying a size

```
#define SBRK_SIZE 2048
```

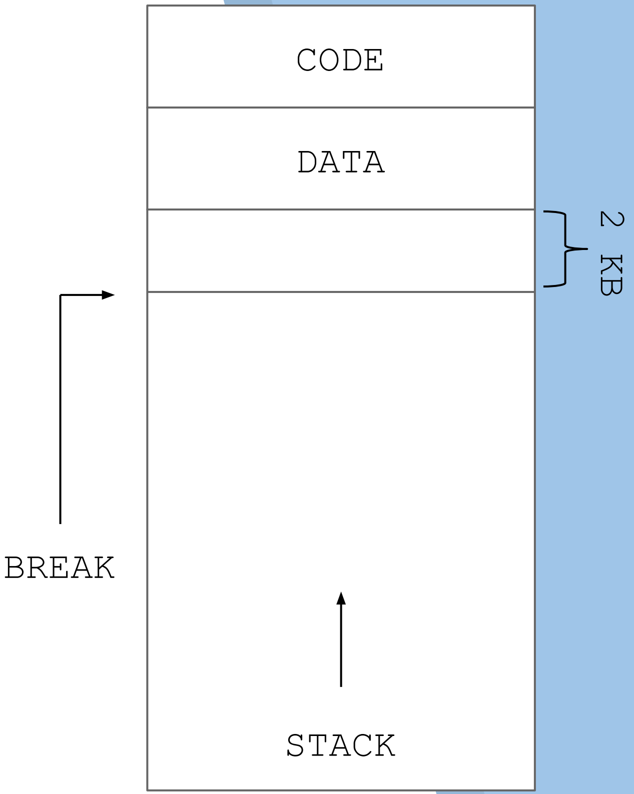


Homework 11

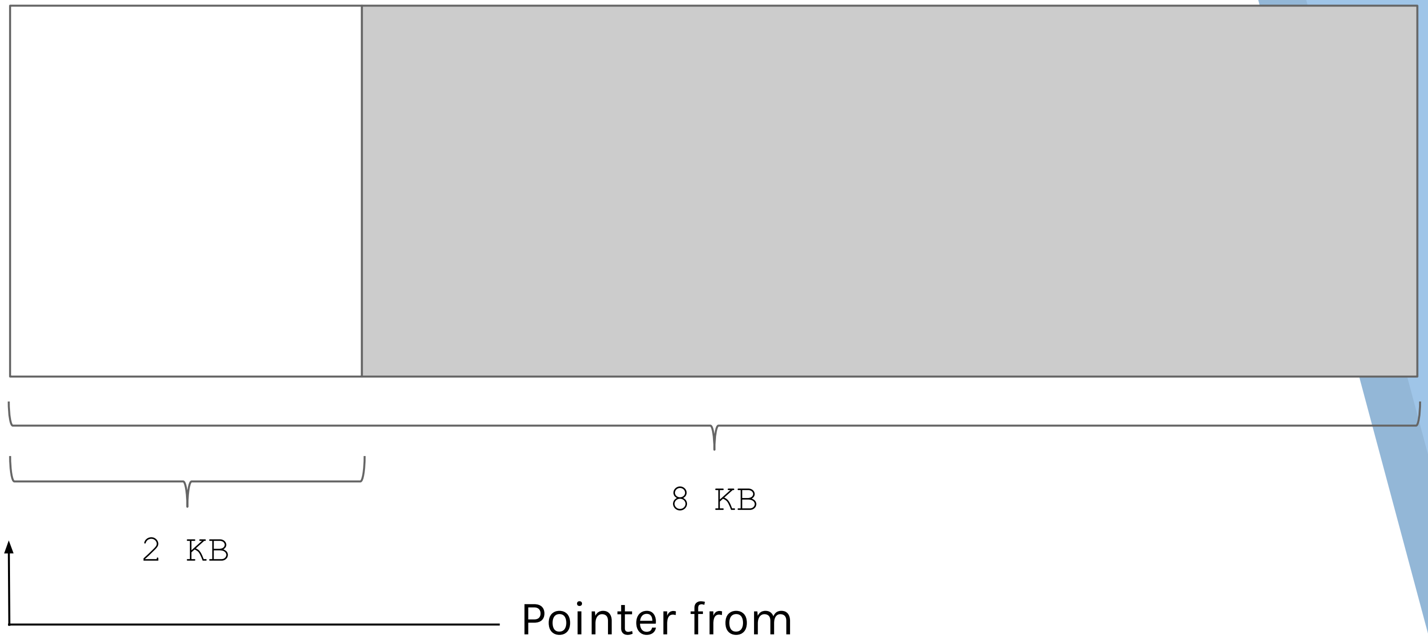
Change end of the process's data segment:

- `brk()` - by specifying an address
- `sbrk()` - by specifying a size

```
#define SBRK_SIZE 2048
sbrk(SBRK_SIZE);
```



Homework 11



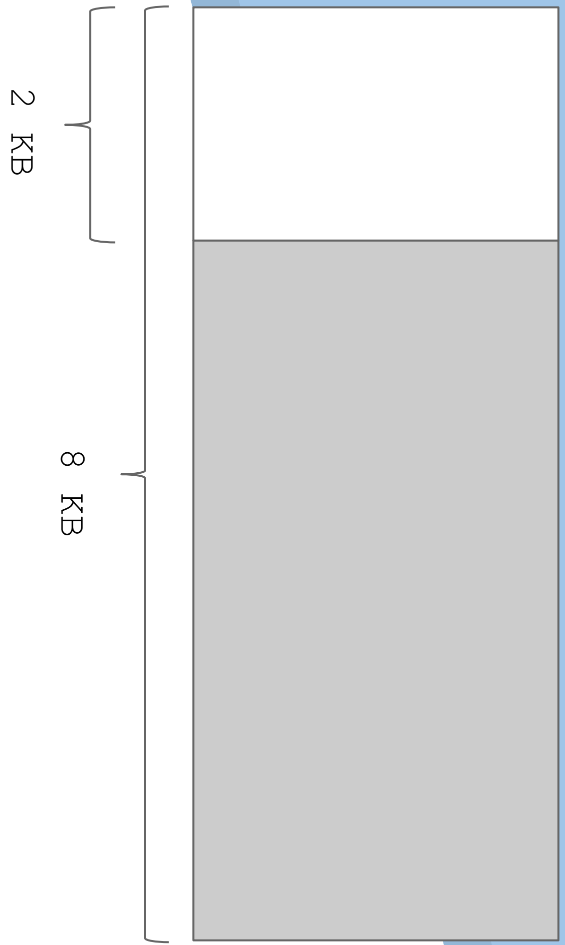
8 KB

2 KB

Pointer from `sbrk`

Homework 11

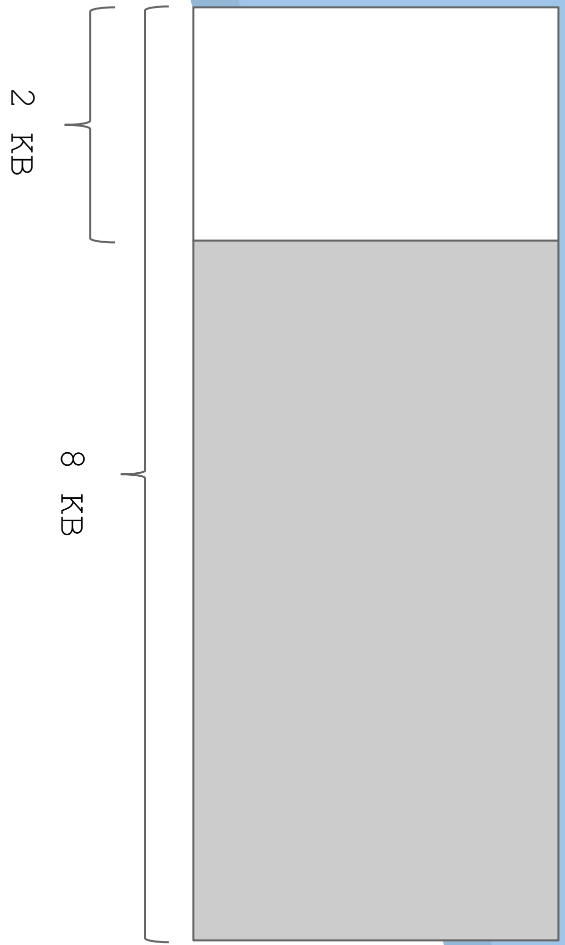
How do we allocate blocks?



Homework 11

How do we allocate blocks?

```
typedef struct metadata
{
  short block_size;
  short request_size;
  struct metadata *prev;
  struct metadata *next;
}
```



Homework 11

What is the freelist?

Homework 11

What is the freelist?

```
struct metadata_t* freelist;
```

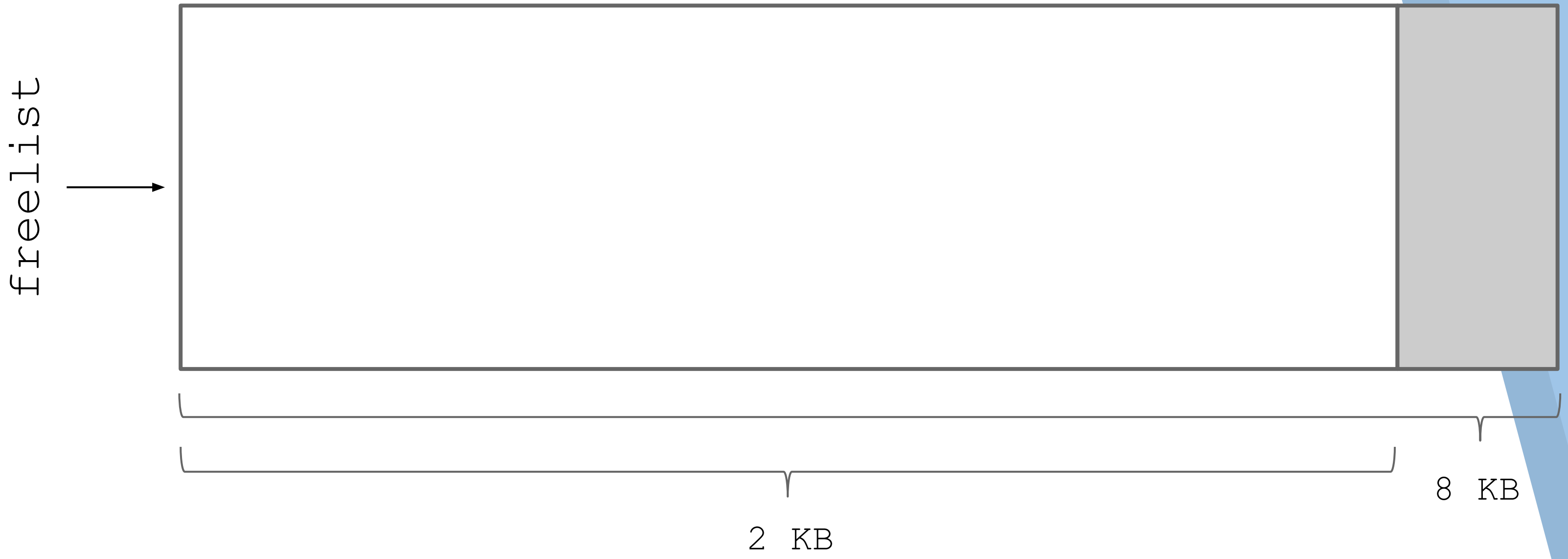
Homework 11

What is the freelist?

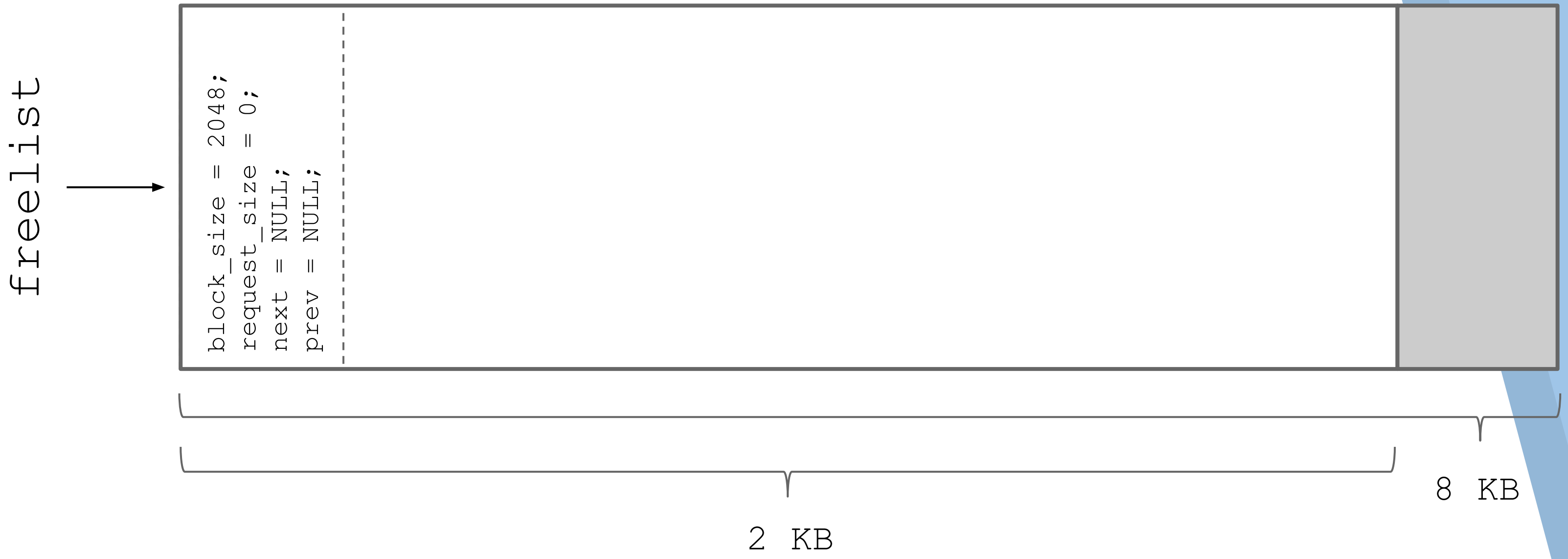
```
struct metadata_t* freelist;
```

Doubly-linked list of unallocated (or free) blocks

Homework 11



Homework 11



freelist

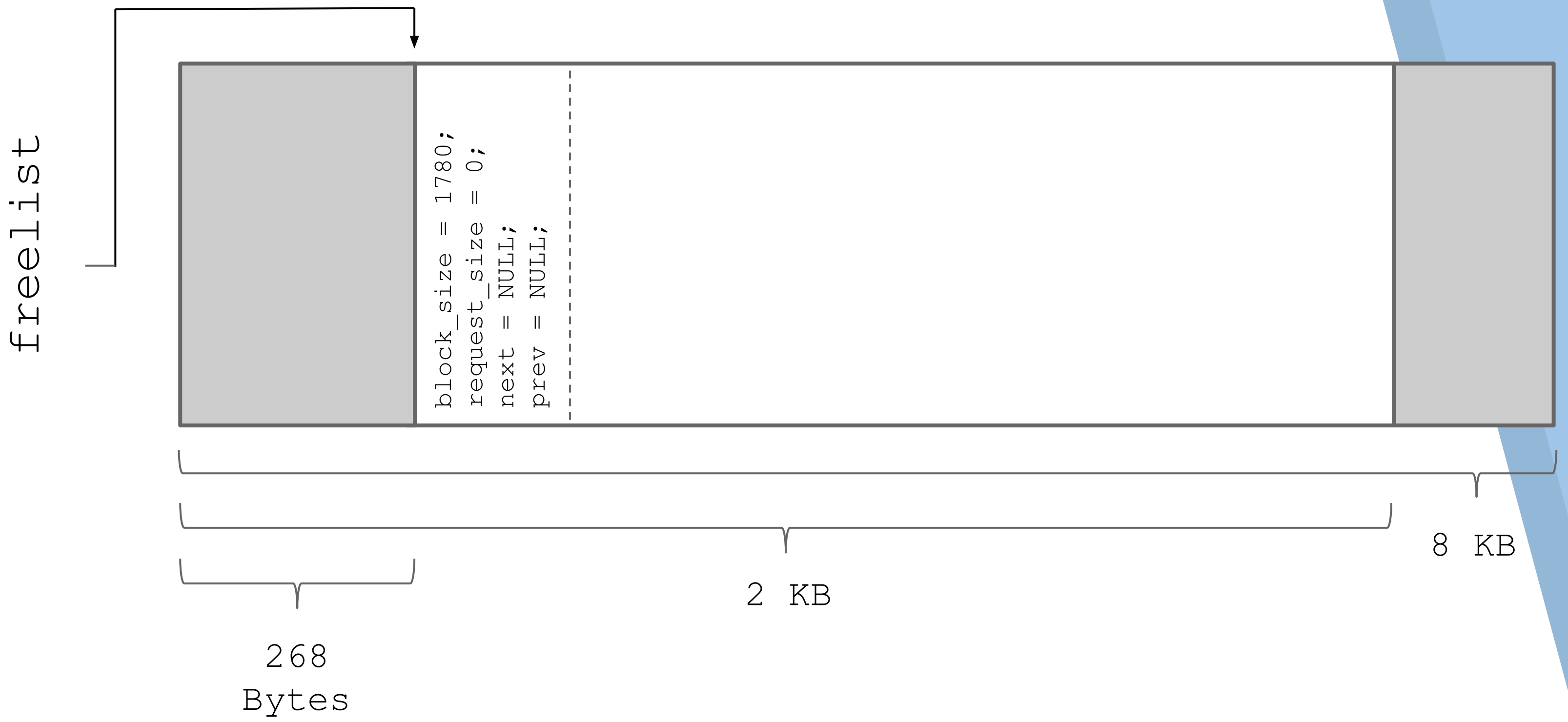
```
block_size = 2048;
request_size = 0;
next = NULL;
prev = NULL;
```

8 KB

2 KB

```
void *a = malloc(256);
```

Homework 11



Homework 11

What if the user overwrites memory in the allocated block?

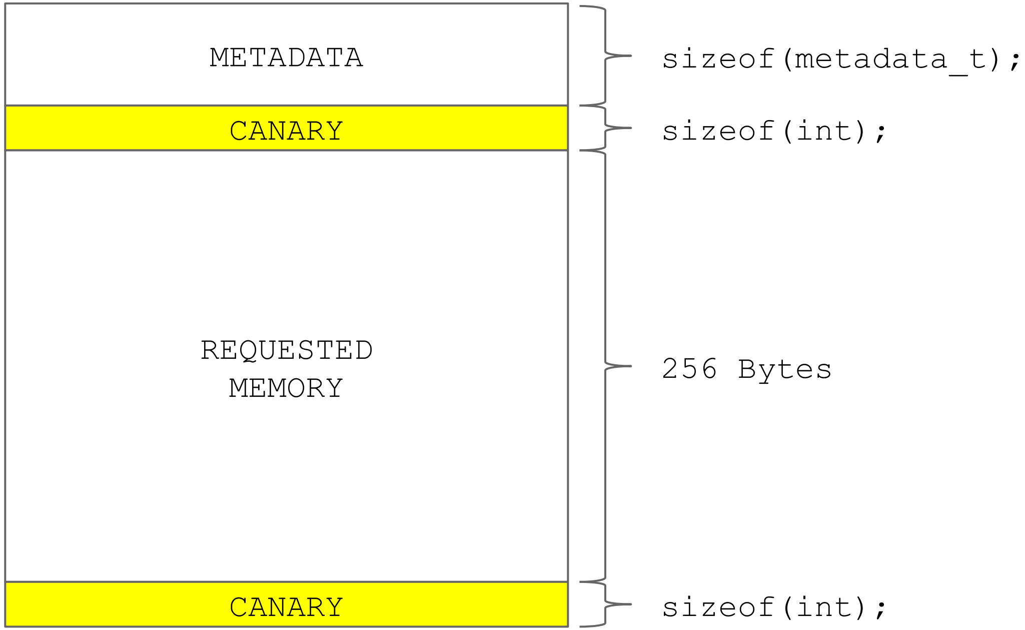
Homework 11

What if the user overwrites memory in the allocated block?

CANARIES!

```
void *a = malloc(256);
```

Homework 11



METADATA

CANARY

REQUESTED MEMORY

CANARY

sizeof(metadata_t);

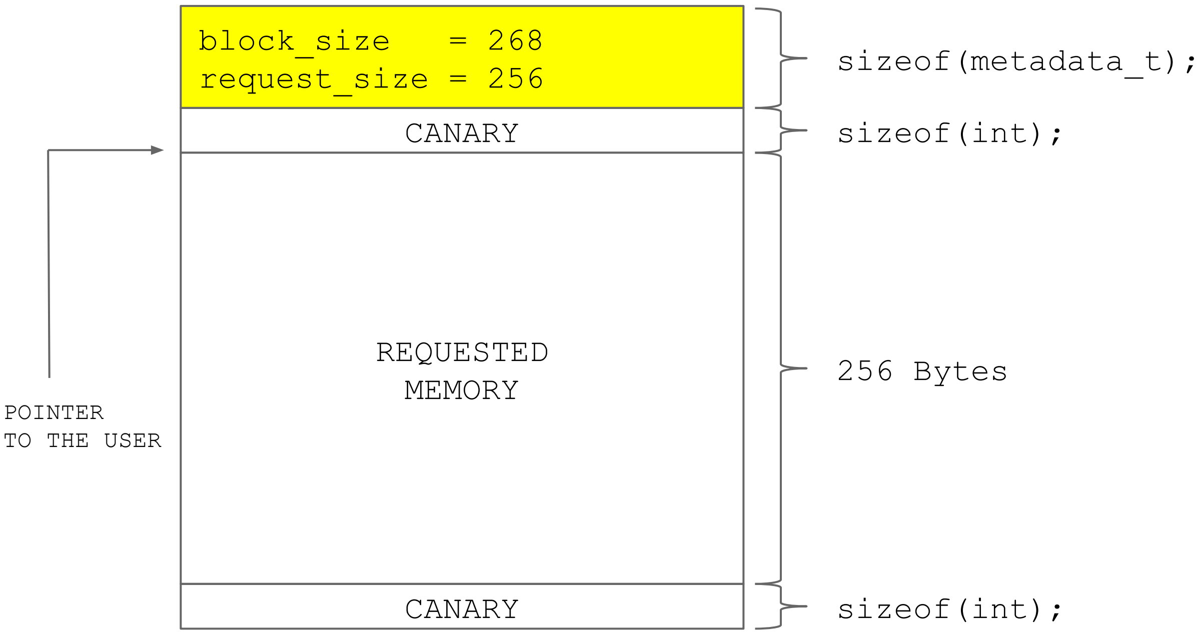
sizeof(int);

256 Bytes

sizeof(int);

Homework 11

```
void *a = malloc(256);
```



```
block_size   = 268
request_size = 256
```

sizeof(metadata_t);

CANARY

sizeof(int);

POINTER
TO THE USER

REQUESTED
MEMORY

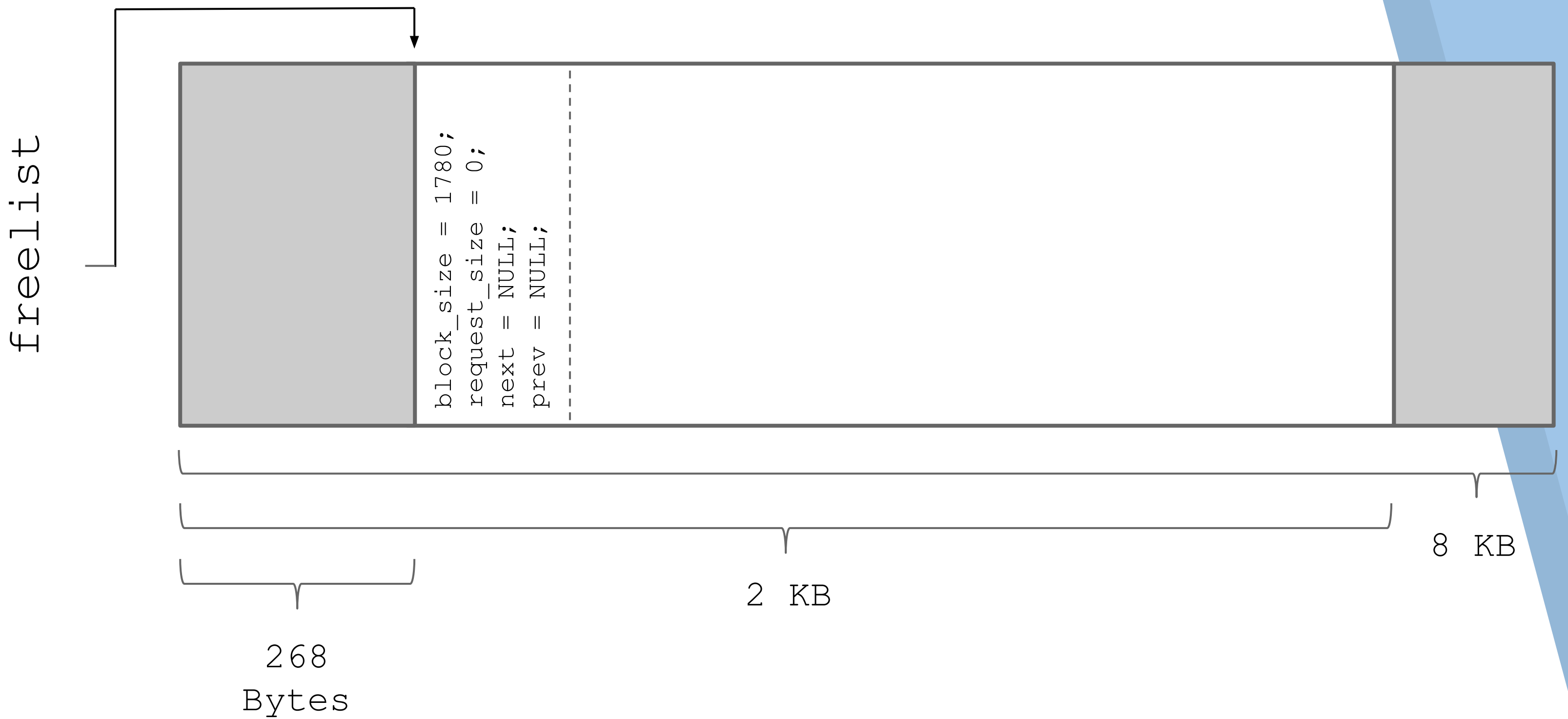
256 Bytes

CANARY

sizeof(int);

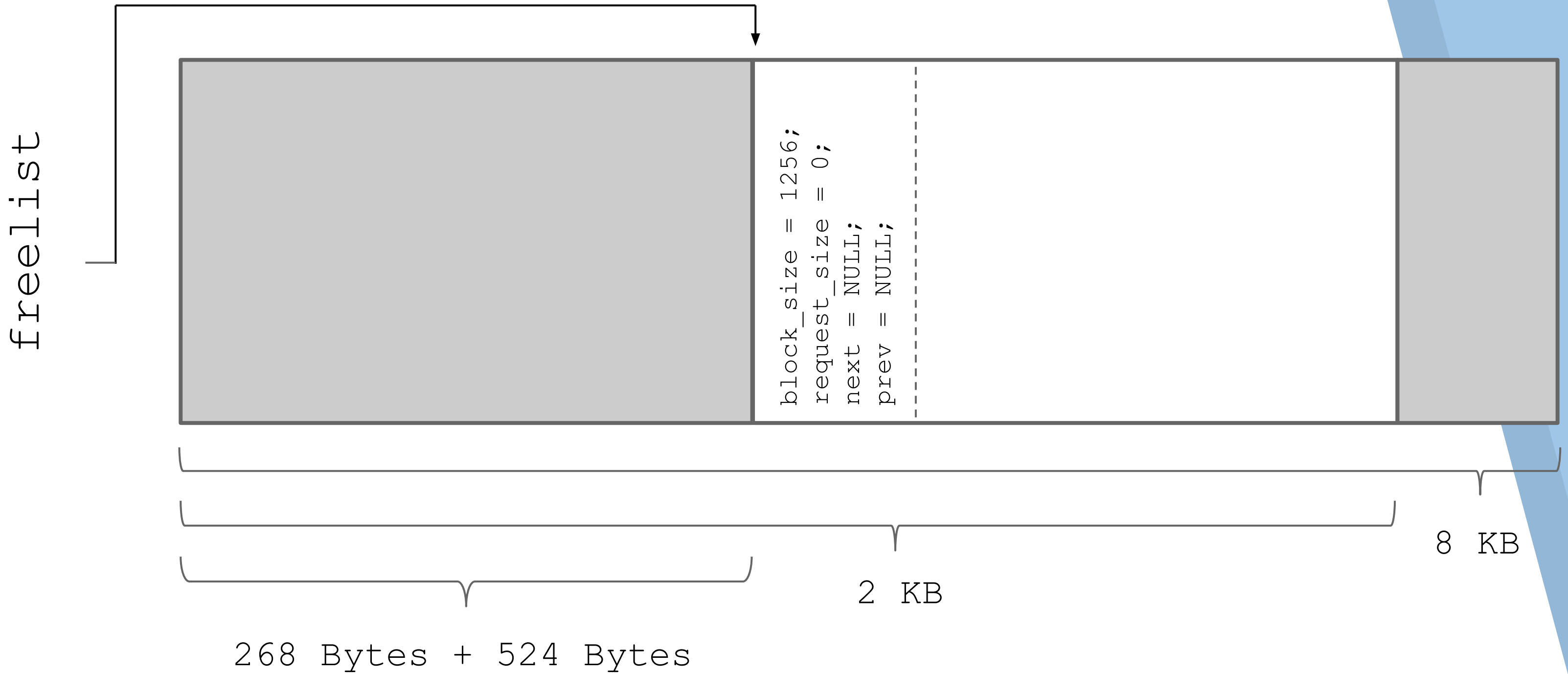
```
void *a = malloc(256);
```

Homework 11



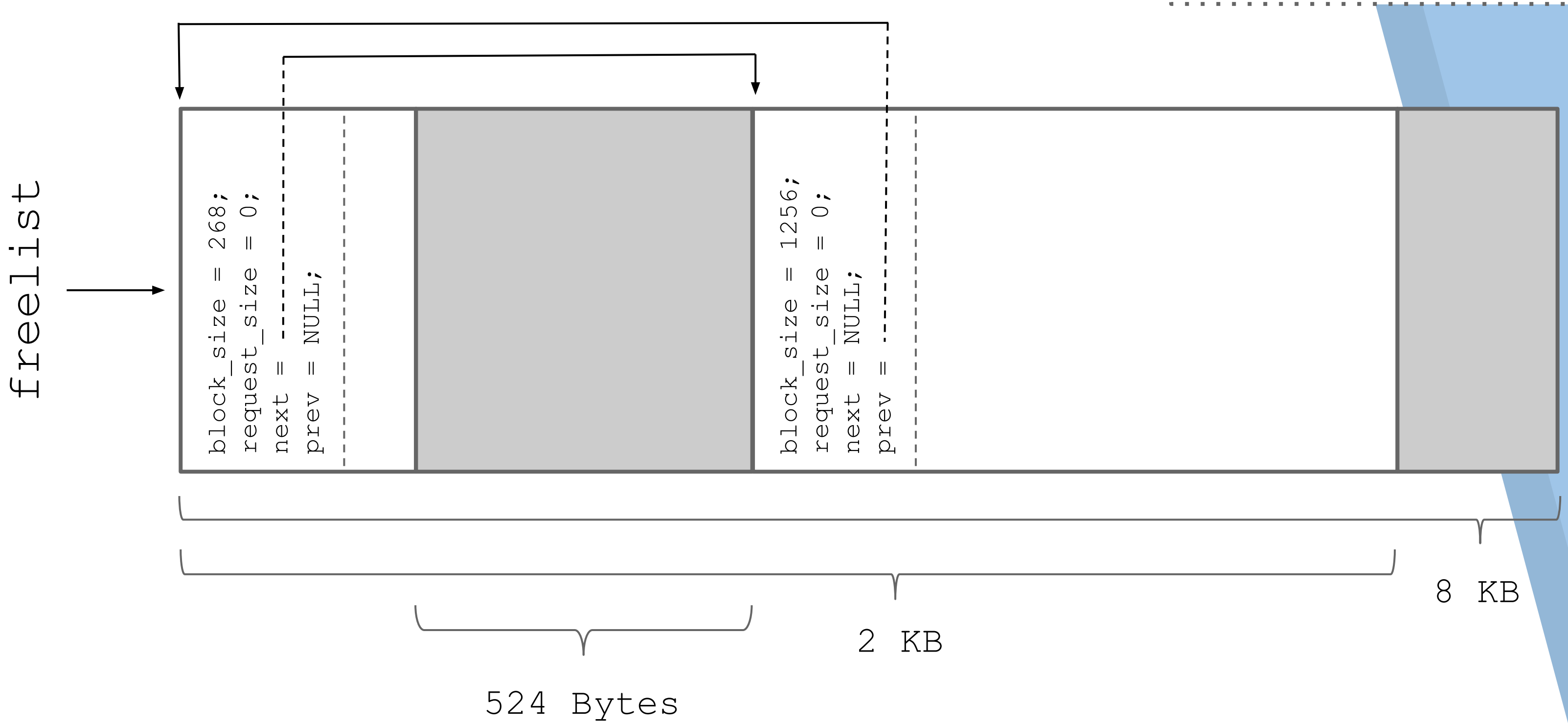
Homework 11

```
void *a = malloc(256);
void *b = malloc(512);
```



```
void *a = malloc(256);
void *b = malloc(512);
free(a);
```

Homework 11



freelist

```
block_size = 268;
request_size = 0;
next =
prev = NULL;
```

```
block_size = 1256;
request_size = 0;
next = NULL;
prev =
```

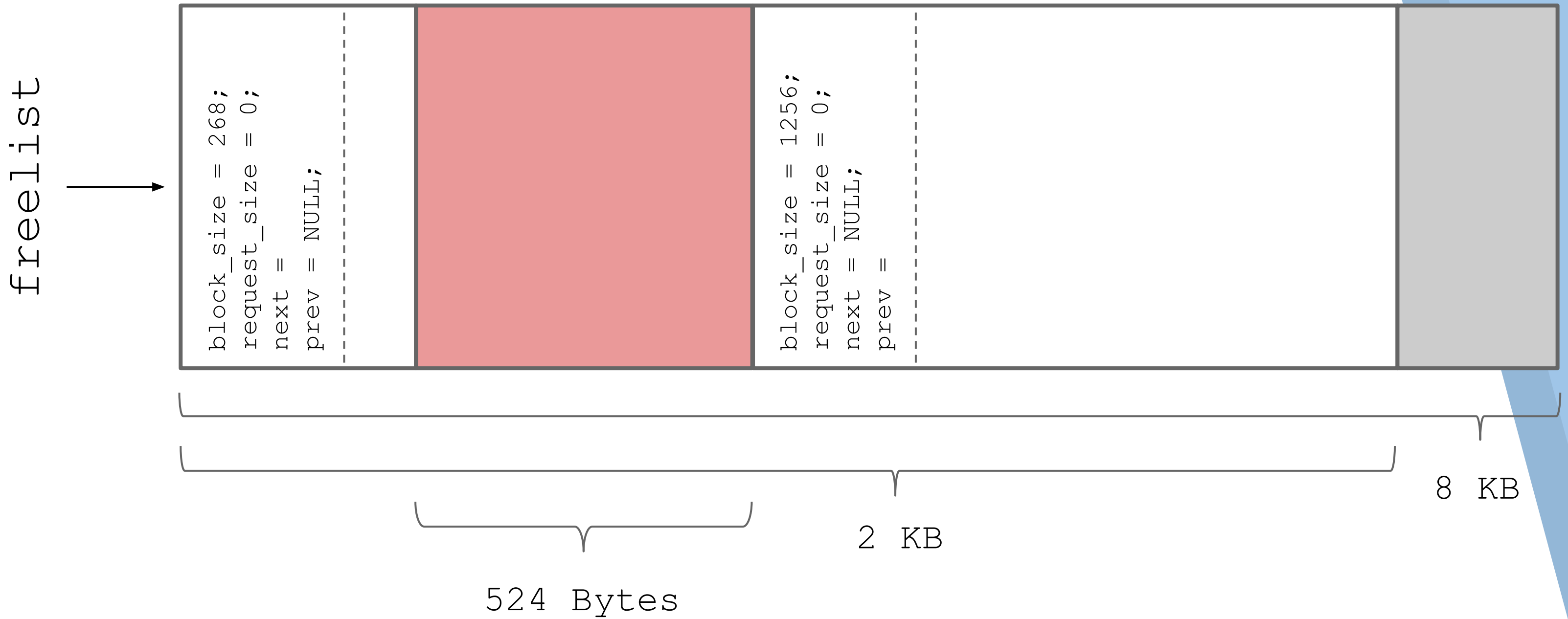
524 Bytes

2 KB

8 KB

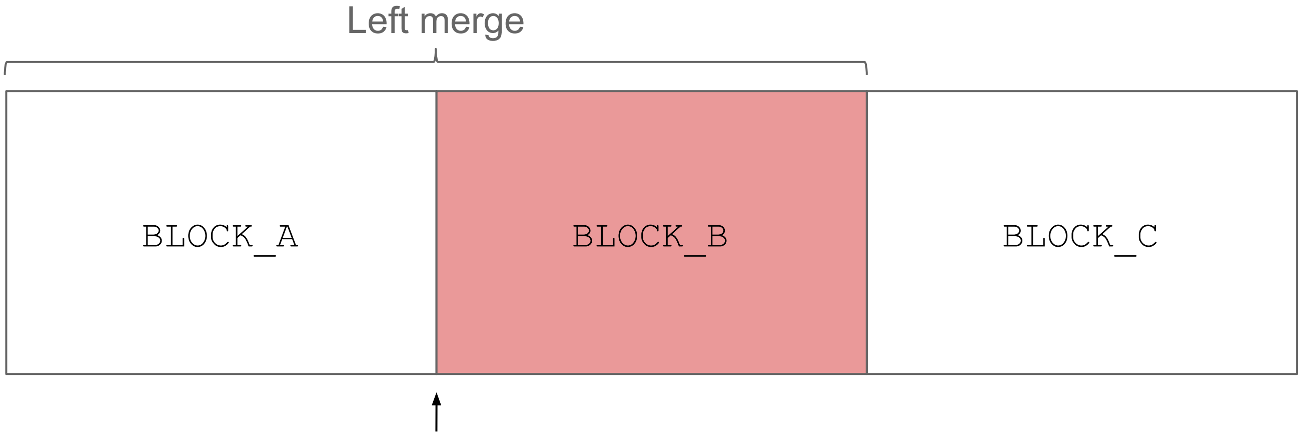
Homework 11

```
void *a = malloc(256);
void *b = malloc(512);
free(a);
free(b);
```



Homework 11

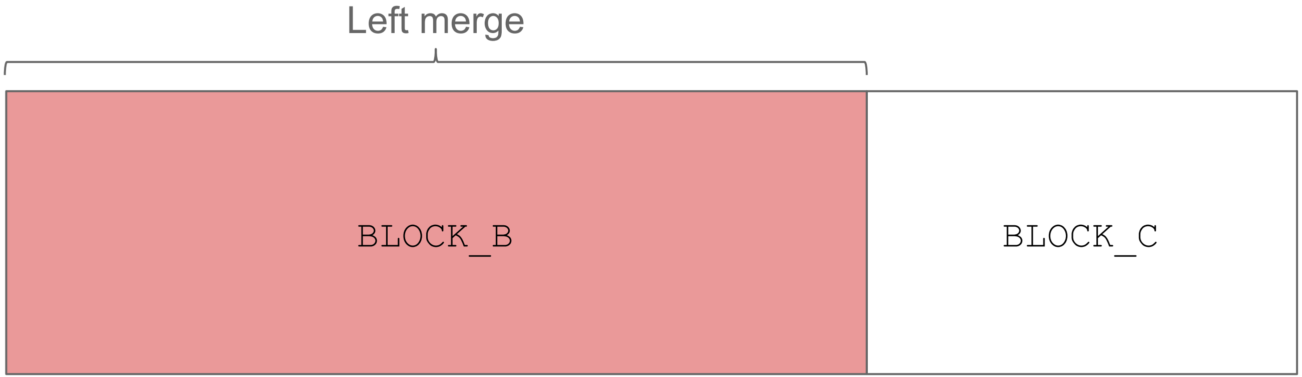
Merging freed blocks:



BLOCK_A address + BLOCK_A.block_size == BLOCK_B address

Homework 11

Merging freed blocks:

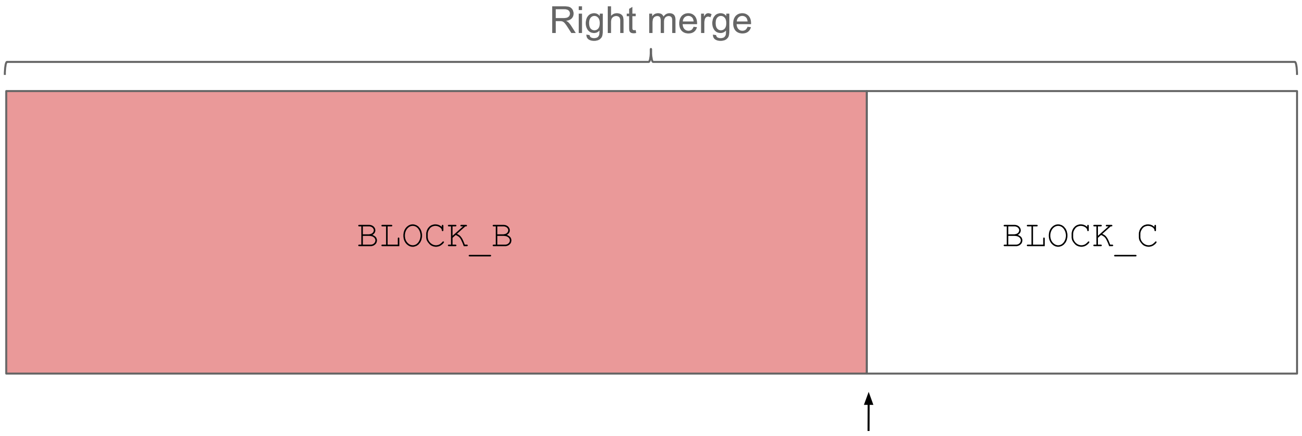


Now has:

```
block_size = BLOCK_A.block_size + BLOCK_B.block_size
address    = BLOCK_A address
```

Homework 11

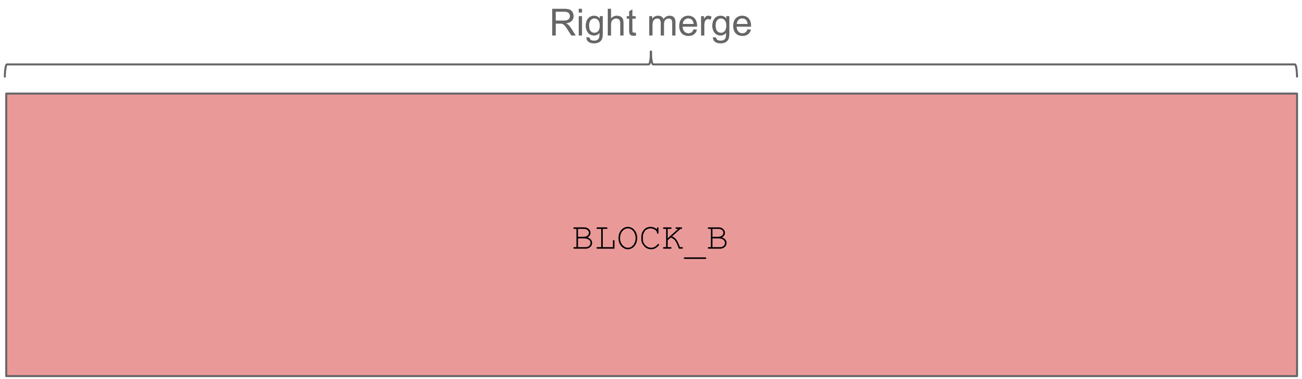
Merging freed blocks:



BLOCK_B address + BLOCK_B.block_size == BLOCK_C address

Homework 11

Merging freed blocks:



Right merge

BLOCK_B

Now has:

```
block_size = BLOCK_B.block_size + BLOCK_C.block_size
address    = same address
```

Homework 11

More things to consider:

Homework 11

More things to consider:

- Minimum block size after split

Homework 11

More things to consider:

- Minimum block size after split

```
sizeof(metadata_t) +
2 * sizeof(int) + 1
```

Homework 11

More things to consider:

- Minimum block size after split

```
sizeof(metadata_t) +
2 * sizeof(int) + 1
```

- Sorted / unsorted freelist

Homework 11

More things to consider:

- Minimum block size after split

  ```
  sizeof(metadata_t) +
  2 * sizeof(int) + 1
  ```

- Sorted / unsorted freelist
  - By address?
  - By size?

Homework 11

More things to consider:

- Minimum block size after split

  ```
  sizeof(metadata_t) +
  2 * sizeof(int) + 1
  ```

- Sorted / unsorted freelist
  - By address?
  - By size?
- `/* Comment your code! */`

Questions?

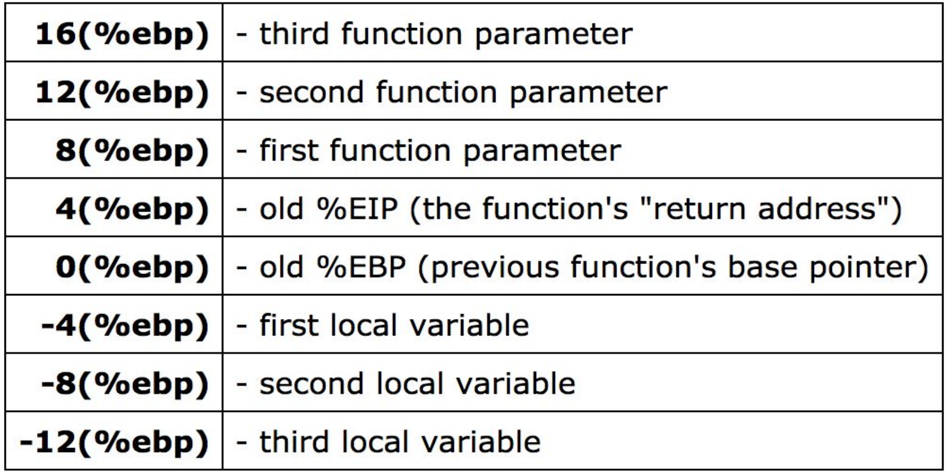
What About Today?

Assignment under "Assignments":

- Download lecitation16.tar.gz on T-Square
- Unlimited submissions
- Be sure you get checked off by a TA

What About Today?

Stack Smashing:



16(%ebp)	- third function parameter
12(%ebp)	- second function parameter
8(%ebp)	- first function parameter
4(%ebp)	- old %EIP (the function's "return address")
0(%ebp)	- old %EBP (previous function's base pointer)
-4(%ebp)	- first local variable
-8(%ebp)	- second local variable
-12(%ebp)	- third local variable

What About Today?

Stack Smashing:

```
$ make
  ...
$ gdb ./hex2ascii
  ...
(gdb) p main
  $1 = {int (void)} 0x4007b2 <main>
```

Questions?