Dynamic memory

Key functions:

```
#include <stdlib.h>
```

malloc(n) allocates n bytes and returns the memory address (type void*) of the first byte.

free(a) deallocates the allocation block beginning at memory address a.

Here is an example of a very simple usage.

```
#include <stdlib.h>

int main(int argc, char** argv) {
   // Allocate memory sufficient for an array of two int's.
   int* a = malloc(sizeof(*a) * 2);  // no error handling, for now

   // Populate the new array.
   a[0] = 768;
   a[1] = 337;

   // Deallocate the allocation block containing the array.
   free(a);

   // Set a to NULL to avoid accidentally reading/writing the allocation
   // block after it has been deallocated.
   a = NULL;

   return EXIT_SUCCESS;
}
```

Fill in the contents of the stack and heap just before the return statement is executed. Assume `sizeof(int) == 4`, `sizeof(char) == 1`, and `sizeof(void*) == 8`. The program was called as "./foo".

If a value is uninitialized or has been freed (aka "garbage"), write the last known value and cross it out.

Stack (simplified)

addr.	*type**	*name**	*value*	*fn*	*part*
212	*somewhere in bash*				return addr.
204	char**	argv	➔ {"./foo"}		parameters
200	int	argc	1		
	bash / load system				

Heap (simplified)

addr.	*pmvalue*
412	
411	
410	
409	
408	
407	
406	
405	
404	
403	
402	
401	
400	

* Note: type and name are not actually stored in memory. They are listed here only for our understanding.

This version has been updated to remove the "name" and "type" column from the Heap and add the note above.