Zack Colello and Anna Genke

Assignment 6: Error detecting malloc() and free()

our implementation of malloc consists of mymalloc.c, mymalloc.h and a makefile that creates an executable ./MallocTest .

Design:

Our program utilizes an array of size 2^{20} . 25% of the block is reserved for allocation of 8 bytes and smaller. The Block is divided into mementry structs. A mementry sturct has a previous, a next, an int isfree of one bit, an int size of 31 bits and an int recognition pattern. Memory is allocated with a first-fit algorithm, in that the mymalloc() function will iterate through the list of memory until it finds the first available space large enough.

To avoid fragmentation, 25% of the memory block is reserved for small allocation. If the size of the requested allocation is 8 bytes or smaller, the malloc function will begin looking for space 75% into the memory block. This way the majority of the smaller allocations are at the end of the memory space and are less likely to cause fragmentation amongst the larger allocations.

All errors are in Christmas colors as per request of BKR.

Function:

void* mymalloc(int size);
-allocates a portion of memory of size bytes, or more.
-if there does not exist enough space for the requested allocation, the program will exit with error message.

void* mycalloc(int num, int size);
-allocates a postion of memory with space for num entries of size bytes.
-if there does not exist enough space for the requested allocation, the program will exit with error message.

void* myrealloc(void* ptr, int size);
-if ptr == Null, it allocates a chunk of memory of size bytes.
-if size == 0, it frees the memory pointed to by ptr.
-otherwise it allocates a chunk of memory of size bytes and copies the content pointed to by ptr into the new chunk, it then frees the old chunk.

void free(void* ptr);
-deallocates the memory allocated by one of the above functions.
The program will exit with an error message under the following circumstances:

- if ptr does not point to allocated memory
- if ptr does not point to the beginning of a chunk of memory that was returned from one of the above functions
- if the ptr points to a chunk of memory that has already been freed

int main();
-used to test the program.
Main contains a switch case to demonstration different types of test for the program

Case 1 - Single char* mymalloc with myfree
Case 2 - Multiple char* mymalloc with myfree
Case 3 - Single char* mycalloc with myfree
Case 4 - Multiple char* with mycalloc, mymalloc and myfree
Case 5 - Single char* with mymalloc and myrealloc
Case 6 - Multiple char* with mymalloc, myrealloc and mycalloc

Error testing Options:
Case 7 - Allocating too large of a size
Case 8 - Attempt to free null variable
Case 9 - Attempt to free something unable to be freed, e.g. bad pointer arithmetic
Case 10 - Attempt to free something that has already been freed