Joshua Lyons & Conner Turnbull
CS361
Lab 03
October 1, 2015
1.) Allocate memory for an array of n pthread_t elements and assign it to variable "threads"

2.) The role of the pthread_create_variable is to provide a location in memory which pthread_attr_init can map to, which allows us to use the address of pthread_create_attr in pthread_create for each thread.

3.) The (p+i) on line 50 is pointer arithmetic, which is being used to access and create each thread in p. The (void *) cast is necessary because the hello function takes a void argument.

4.) The threads do not always run in the same order. There is no way to tell what order each thread will finish running, but when each finishes, it prints "Hello from node xx".

5.) The effect of "pthread_join" not surprisingly joins all the threads that had previously been created. As the threads can finish in ANY specific order, then yes, you could rewrite the statement on line 54 and still get the same effect.

```
turnbucc@L14353:~/cs361/labs/lab03$ gcc p_hello.c -o p_hello.o -pthread
turnbucc@L14353:~/cs361/labs/lab03$ ./p_hello.o 20
Hello from node 0
Hello from node 1
Hello from node 3
Hello from node 7
Hello from node 4
Hello from node 9
Hello from node 6
Hello from node 10
Hello from node 2
Hello from node 5
Hello from node 8
Hello from node 11
Hello from node 12
Hello from node 13
Hello from node 14
Hello from node 15
Hello from node 16
Hello from node 17
Hello from node 18
Hello from node 19
turnbucc@L14353:~/cs361/labs/lab03$ ./p_hello.o 20
Hello from node 1
Hello from node 3
Hello from node 0
Hello from node 2
Hello from node 6
Hello from node 4
Hello from node 5
Hello from node 7
Hello from node 8
Hello from node 9
Hello from node 10
Hello from node 11
Hello from node 12
Hello from node 13
Hello from node 14
Hello from node 15
Hello from node 16
Hello from node 17
Hello from node 18
Hello from node 19
turnbucc@L14353:~/cs361/labs/lab03$
```