Brandon Fowler

lab5prob 1 output

```
brandonfowler@Server-Ubuntu:~/Desktop$ ./prob1
Creating thread 1
Creating thread 2
Creating thread 3
Hello thread 2
Creating thread 4
Goodbye thread 1
Creating thread 5
Goodbye thread 3
Creating thread 6
Hello thread 4
Goodbye thread 5
Hello thread 6
brandonfowler@Server-Ubuntu:~/Desktop$ ./prob1
Creating thread 1
Creating thread 2
Creating thread 3
Goodbye thread 1
Goodbye thread 3
Creating thread 4
Hello thread 2
Creating thread 5
Hello thread 4
Creating thread 6
Goodbye thread 5
Hello thread 6
brandonfowler@Server-Ubuntu:~/Desktop$ ./prob1
Creating thread 1
Creating thread 2
Goodbye thread 1
Creating thread 3
Hello thread 2
Creating thread 4
Goodbye thread 3
Creating thread 5
Hello thread 4
Creating thread 6
Goodbye thread 5
Hello thread 6
brandonfowler@Server-Ubuntu:~/Desktop$ ./prob1
Creating thread 1
Creating thread 2
Creating thread 3
Goodbye thread 1
Creating thread 4
Goodbye thread 3
```

Creating thread 5
Hello thread 2
Hello thread 4
Creating thread 6
Goodbye thread 5
Hello thread 6

lab5prob2 output

Thread 1 filling array element 0 with 100
Thread 1 filling array element 1 with 101
Thread 1 filling array element 2 with 102
Thread 1 filling array element 3 with 103
Thread 1 filling array element 4 with 104
Thread 1 filling array element 5 with 105
Thread 1 filling array element 6 with 106
Thread 1 filling array element 7 with 107
Thread 1 filling array element 8 with 108
Thread 1 filling array element 9 with 109
Thread 2 filling array element 0 with 100
Thread 0 filling array element 0 with 100
Thread 3 filling array element 0 with 100
At index 0: 100
At index 1: 101
At index 2: 102
At index 3: 103
At index 4: 104
At index 5: 105
At index 6: 106
At index 7: 107
At index 8: 108
At index 9: 109

lab5prob2 questions

a) tid is an array of pointers to pthread_t variables. The value it contains at a given location(tid[x]), is an id that identifies the thread within the program. This also allows us to use operations on individual threads such as join, since we can identify each thread by the id's stored in tid.

b) Using pthread_join, insures that we will receive the full output from each thread that is joined. If no joins are used, each thread will still run but the main program may end before they complete their tasks. Any output from threads that end after the main program ends will be trapped on the buffer, and ultimately lost.

c) It seems that in this example the call to pthread_join may not be entirely necessary. Since the final array is being printed, the threads should have time to finish, and output will still be printed to the screen. However, the join calls should still be used so that output is arranged in the correct order, and also to provide a guarantee that no output is lost.

d) The tid in the call to pthread_join, identifies exactly what thread within the program that we wish to join.

e) The tid should be the tid of the current thread that is being joined, not just any tid.