1: Run the program produced by compiling pthread test.c. What is expected output of the program? Explain the output.

The expected output is 10 threads being created inside a loop, a message being printed out informing that thread number X has been created from the main() and a function being called for each thread creation where the function prints out a message containing the thread id (fake/self created id and not the actual thread id) and terminating the thread with pthread_exit. If the thread creation fails an error message should be printed out. Lastly the thread (which in this case is the main thread) should be terminated with pthread_exit.
ret = pthread_create(&threads[i], NULL, PrintMessage, (void *)i); this is the part where the thread is created. The first argument is referring to the address of the threads, which is a pthread_t variable, and we have [i] because we have multiple threads since they're being created inside the loop. The second argument is NULL, meaning that the thread attributes are set to default values. PrintMessage is a function call to the function, and here it is important to note that it should be a pointer to a void function; the function either has to have * in front of it's name at the time of creation, or it has to be called like (void*)PrintMessage to typecast. The last argument casts the variable i to (void*) because this is the only datatype accepted, but it can also be a nullpointer if the function doesn't require parameters.

2: Explain the output. How the output of this program is different from that of Exercise 1.

The only two differences in the code is that the function sleeps for 2 seconds before executing which in this case is exiting the function, instead of terminating the thread right in the first task. pthread_exit can be omitted from the function because returning from the function of the thread performs an implicit call to pthread_exit(), and it's called regardless of how you terminate your thread. It's also responsible for thread's cleanup.
The other difference is that pthread_join is added at the end of the loop to join the caller thread (main in this case) with the threads created; this is done to make sure that the caller thread waits for each and all of the joined threads to terminate before continuing executing.
The output is different because sleep and join makes sure that the the threads and the corresponding function calls are created and executed in order, instead of the inconsistencies of task one where we first have two thread creations before the first function call, and end with two consecutive function calls.

3.2 What is the output of the program? Does it match with the expected output of the program?

Expected to print I am thread one from function 1 after creation of the first thread, I am thread 2 after the creation of the second thread and finally I am thread zero from the main thread before returning 0. The output is just I am thread zero and return 0 because the main thread is not joined with t1 and t2 and doesn't wait for them to be finished before executing its own code.

3.3 What is the output of the program? Does it match with the expected output of the program?

Each thread starting after it's creation and returning before the creation of another one. Yes, it matches the expected output thanks to pthread_join inside the loop.