```
----------------
PL2 Class Script
----------------

- Thread management; its importance and how it works in Linux
    - heap
- System calls to manage threads
    - pthread_create() - creates a new thread
        - arguments can be passed to the threads
    - pthread_self()   - returns ID of the calling thread
    - pthread_exit()   - terminates the calling thread
    - pthread_join()   - waits for a specific thread and blocks the thread that
calls this function
        - any thread can wait for any thread

- Example Code

#include <stdio.h>
#include <pthread.h>

#define NR_THREADS 5

void* thread_function(void *arg)
{
    printf("running thread: %lu\n", pthread_self());
    pthread_exit((void*)NULL);
}

int main()
{
    pthread_t threads[NR_THREADS];
    int i;

    for(i = 0; i < NR_THREADS; i++)
        pthread_create(&threads[i], NULL, thread_function, NULL);

    printf("All threads were created\n");

    for(i=0;i<NR_THREADS;i++)
        pthread_join(threads[i], NULL);

    printf("All threads finished\n");
}

- Compile

    $ gcc -o output_file input_file -lpthread

- Passing arguments to threads

#include <stdio.h>
#include <pthread.h>
#include <string.h>

#define NR_THREADS 5

void* thread_function(void *arg)
{
    char* received_arg = (char *) arg;

    printf("running thread: %lu received: %s\n", pthread_self(), received_arg);
    pthread_exit((void*)NULL);
}
```

```
int main()
{
    pthread_t threads[NR_THREADS];
    int i;

    char str[8] = "";

    for(i = 0; i < NR_THREADS; i++) {
        snprintf(str, sizeof(str), "MESCC_%d", i);
        pthread_create(&threads[i], NULL, thread_function, (void *) &str);
    }

    printf("All threads were created\n");

    for(i=0;i<NR_THREADS;i++)
        pthread_join(threads[i], NULL);

    printf("All threads finished\n");
}

/* NOTE: Do you notice anything strange with the above code? If so, what? */
```

Exercises
----

1. Develop a program that:

    - creates two arrays, one with 1000 positions and another with 10 positions;
    - creates 10 threads with the responsibility of finding the local maximum in 1/10 of the largest array and store it in the smallest array;
    - the main thread should wait for all the threads to terminate and find the maximum value in the array and print it on the screen.

2. Develop a program that does the same as the program developed in 1. but
    - passes 1/10 of the array as a parameter to the respective thread, and
    - returns the maximum value to the main thread using pthread_exit().

3. Develop a program that multiplies two matrices, considering the following assumptions:
    - The size of each matrix is 16x16;
    - create two threads to fill the two matrices with random integers;
    - create eight threads to perform the multiplication of the matrices;
    - the main thread must wait for all threads to terminate and print the resulting matrix on the terminal.

4. Develop a program that performs the following actions:
    - creates an array of 100 clients;
    - each client is identified by its own number, name and current balance (values are ramdomly generated);
    - creates three threads where:
        - one of the threads verifies if any of the clients has a negative balance and signal each of these;
        - another thread prints the information of the clients with negative balance using the information from the previous thread;
        - another thread computes the average balance of all clients.
    - If the average balance is negative, then another thread must be created to eliminate the negative values;
    - If the average balance is positive, then the main thread should print a statement of conformity and exit.