Question 4: Are the values of ctr printed out by the child threads correct? Explain why or why not.

Answer: Running the program as per usual (./lab3p2), the ctr printed by each child is not correct. In fact, as oppose to the threads printing “I am child X” which prints every child number from 0-9 (but in the wrong order), the ctr is not complete in that sense. Meaning that each number from 0-9 is not printed once. However they are printed in an increasing order (e.g. 0-0-0-0-3-7-7-8-9), ending with “Value of ctr = 10”. The reason for the printed ctr:s by the children is that the variable ctr is a **global variable**. It seems like the scheduling makes the prints of “I am child…” being printed in a faster pace than the ctr++ is run. Also, when the printf (“I am child…”) command is scheduled as a process to be run, the **current** ctr (the value of ctr as it is scheduled) is the one that is sent in as the argument. Thereafter it is the scheduling that decides what ctr the following children will have. A sensed pattern in the ctr values printed is that there are many 0:s and then a jump to e.g. 5. What has happened in between there is that the ctr++ has not been running in between 0 and 5, but rather the prints of “I am child…” has been scheduled and executed in between. This was then followed by ctr++ 5 times before the next print “I am child…” was executed.

*scheduled above refers to scheduled by the OS scheduler

Question 6: Cut and paste your code to you answer book, and explain what pthread_detach does.

Answer: pthread_detach detaches the thread. In opposite of pthread_join, it does not wait for the thread (in parameter thread) to finish (terminate), and for it to be joined with other threads. Instead, as the thread exits its memory is cleaned up and released without another thread being able or needing to join.

Edits of lab3p3.c:

```c
// Maximum number of threads allowed
#define THREAD_LEN  10
```

```c
// Global thread
pthread_t thread[THREAD_LEN];

// Global counter
int ctr = 0;
```

```c
void *child(void *t)
{
    // Increase the thread counter
    ctr++;

    // Deliver the connection
    deliverHTTP((long)t);

    // Exit the thread
    pthread_exit(NULL);
}
```

```c
        while(1)
        {
                connfd = accept(listenfd, (struct sockaddr *) NULL, NULL);
                writeLog("Connection received.");

                // Check if the number of maximum threads is exceeded before threading
                if(ctr < THREAD_LEN)
                    {
                        int currentCtr = ctr;

                        // Start the thread
                        pthread_create(&thread[currentCtr], NULL, child, (void *) connfd);

                        // Detach the thread
                        pthread_detach(thread[currentCtr]);
                    }
        }
}
```