Esercizio 1

(10 min.)

Scrivere un programma per Windows/Unix che permette al processo principale **P** di create un nuovo thread **T** il cui percorso di esecuzione è associato alla funzione "*thread_function*".

Il processo principale **P** ed il nuovo thread **T** dovranno stampare ad output una stringa che li identifichi rispettando l'ordine **T**→**P**, senza utilizzare "*WaitForSingleObject*"/"*pthread_join*", ma sfruttando un concetto fondamentale che accomuna tutti i threads di un determinato processo.

Esercizio 2

(10 min.)

Scrivere un programma per Unix che sia in grado di generare due Thread **T_1** e **T_2**, tali per cui:

- **T_1** chiede all'utente di inserire una messaggio da tastiera

- **T_2** scrive il messaggio ottenuto dall'utente a schermo

con la condizione che non si possono utilizzare variabili globali.

Esercizio 3

(15 min.)

Scrivere un programma per Windows il cui processo principale genera **N** threads.

Ogni thread, con indice **i**, chiede all'utente di inserire un messaggio per il thread con indice **i+1**.

Ogni thread, con indice **i**, come prima cosa stampa il messaggio ricevuto dal thread con indice **i-1**.

Le operazioni di lettura del messaggio ricevuto dal thread precedente, stampa del messaggio e richiesta di un nuovo messaggio per il thread successivo, devono essere sequenzializzate.