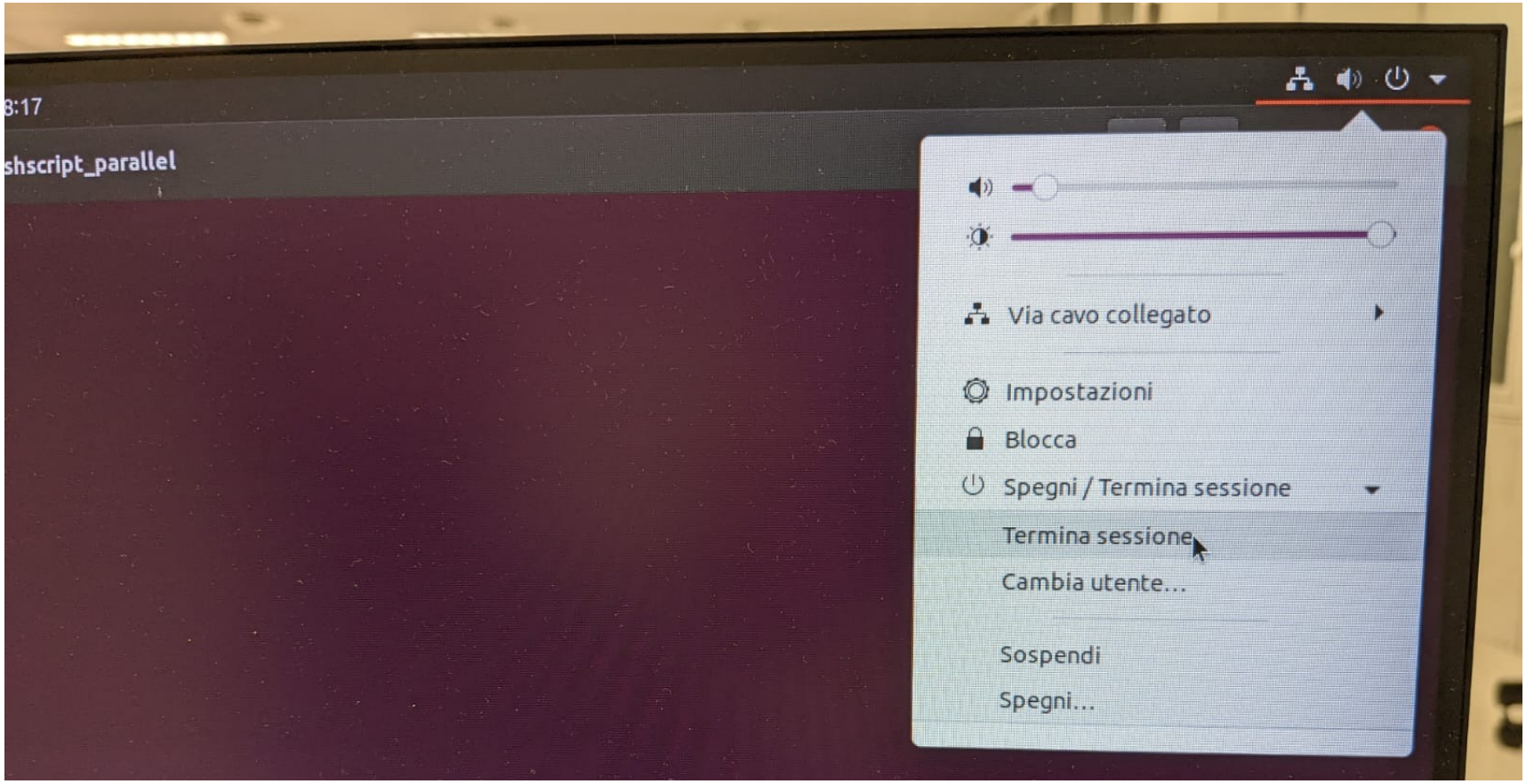
Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta - 19 settembre 2023

ISTRUZIONI:

1. **Rinomina** subito la cartella chiamata "Cognome-Nome-Matricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali e **lasciando i trattini senza spazi**; se hai un doppio nome oppure un doppio cognome dovrai chiamare la cartella come in questo esempio:
    a. DeLuca-MarcoGiovanni-199999
2. **Carica** tutto il materiale didattico che vorrai usare sul Desktop; puoi farlo solo nei primi 5 minuti della prova;
3. **Svolgi** il compito; lascia tutto il sorgente che hai prodotto nella cartella di cui al punto 1;
4. Quando hai finito fai "**Termina Sessione/Logout**", ma lascia la postazione mantenendo il PC acceso.
5. **E' tua diretta responsabilità** garantire l'integrità del tuo elaborato, anche in caso di assenza di corrente. **Salva spesso il tuo lavoro**



e NON spegnere il PC.

SALVA SPESSO

CI SONO DEI PUNTI AMBIGUI NELLA TRACCIA? COMPLETA TU

È parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo nuove strutture dati, o estendendo quelle preesistenti laddove si ritenga necessario, risolvendo eventuali ambiguità. Si può cambiare il codice dei metodi esistenti dove serve.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI O DI QUELLI ESISTENTI? NO

Non è consentito modificare il prototipo dei metodi se questo è stato fornito. Potete aggiungere qualsivoglia campo e metodo di servizio, e qualsivoglia classe ausiliaria, ma NON variare l'interfaccia dei metodi pubblici già specificati. Analogamente, i metodi esistenti possono essere modificati nel loro codice, ma non se ne deve cambiare il risultato finale o il significato.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X

Il linguaggio da utilizzare per l'implementazione è Python 3.6 o successivo. Ricorda che l'operatore di formattazione `f` (esempio, `f"Ciao sono la stringa {testo}"`) è disponibile solo dalla versione 3.6 di Python in poi, ma può essere sostituito con `"Ciao sono la stringa %s" % testo`

POSSO CONSENTIRE SITUAZIONI DI RACE CONDITION NEL MIO CODICE? NO
POSSO CONSENTIRE SITUAZIONI DI DEADLOCK NEL MIO CODICE? NO
POSSO CONSENTIRE ALTRE SITUAZIONI DI BLOCCO TOTALE NEL MIO CODICE, TIPO NESTED LOCKOUT, LIVELOCK O ALTRO? NO
POSSO CONSENTIRE SITUAZIONI DI STARVATION NEL MIO CODICE? SI, tranne quando ti viene chiesto esplicitamente di rimuoverle

MA IL MAIN() LO DEVO AGGIORNARE? E I THREAD DI PROVA? SI

E' obbligatorio implementare esplicitamente il codice di prova oppure modificare il codice di prova pre-esistente, e accertarsi che giri senza errori prima della consegna.

ESERCIZIO 1 - PROGRAMMAZIONE MULTITHREADED

(Punteggio minimo richiesto 18/30. Pesa per ⅔ del voto finale)

Per questa prova di esame, dovrai progettare la classe **Ristorante** che combina una istanza di Pizzeria con una istanza di classe **Sala** e con altri elementi che ora ti descriverò. Una *Sala* è composta da N tavoli da 10 posti ciascuno. Ciascun posto può essere vuoto oppure essere occupato da una pizza. La Sala comunica con la pizzeria sostituendo ai Clienti, delle istanze di thread *Cameriere*.

Il Cameriere è un nuovo tipo di thread che dovrai progettare appositamente. Così come avviene nel codice esistente di *Cliente,* un *Cameriere* genera un ordine casualmente, lo affida ai pizzaioli usando il metodo `putOrdine` per poi prelevare le pizze con il metodo `getPizze`**.**

Ci sono tuttavia delle differenze rispetto ai clienti:
-ogni ordine ora dovrà indicare un tavolo designato T, scelto tra quelli completamente sgombri, e appartenente alla Sala; T rappresenta il tavolo dove il cameriere dovrà depositare le pizze appena sfornate;
-gli ordini non possono superare la quantità di dieci pizze totali;
-quando l'ordine è pronto, il cameriere sistema le pizze sul tavolo T, ma poichè può trasportare al massimo due pizze per volta, è costretto a fare più viaggi. Sarà tuo compito modificare il codice per fare in modo che si possano prelevare da uno specifico ordine contenuto nel buffer delle pizze *2 elementi alla volta al massimo*; inoltre, tra un viaggio e l'altro dovrai introdurre del tempo di attesa che simula il tempo che ci vuole ad andare dalla Pizzeria alla Sala e viceversa.

Una sala è inoltre dotata di una o più istanze di thread *Sparecchiatore.* Uno sparecchiatore toglie periodicamente le pizze dai tavoli della Sala, ma dovrai assicurarti che una pizza appena depositata da un *Cameriere* non sia rimossa prima di 3 secondi dal momento in cui è stata poggiata sul tavolo.
Questo è tutto. Come sempre, è tuo specifico compito decidere cosa fare per tutti i dettagli che non sono stati espressamente definiti.

SALVA SPESSO

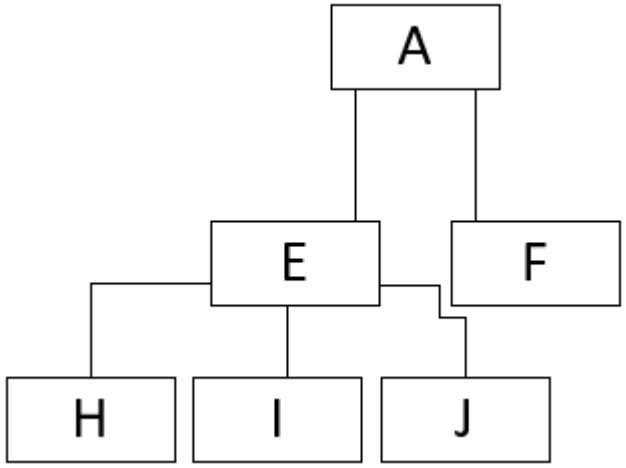
ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

(Punteggio minimo richiesto 18/30. Pesa per ⅓ del voto finale)

Scrivi uno script perl dal nome `cerca_e_copia.pl`. Lo script dovrà comportarsi nel seguente modo:

1. se eseguito senza opzioni, lo script dovrà:
    1.1. cercare tutti i file con estensione ".pl" che si trovano direttamente all'interno di qualche cartella allo stesso livello di quella da cui è stato lanciato lo script `cerca_e_copia.pl`. N.B. ricorda che la cartella dalla quale viene lanciato lo script non è sempre quella in cui si trova lo script stesso. **Ad esempio**, dati gli alberi di cartelle sottostanti, supponiamo che lo script `cerca_e_copia.pl` si trovi in `E`. Se lo script viene invocato dalla cartella `E` allora i file ".pl", l'unica cartella di pari livello di `E` è `F`, dunque i file devono essere cercati SOLO in `E` e `F`; se lo script viene invocato da `J`, le cartelle di pari livello di `J` sono `I` e `H`, dunque bisognerà cercare

SOLO in `H`, `I` e `J`.



1.2. copiare i file trovati in un cartella che abbia come nome la tua matricola e che verrà creata dallo script nella cartella dalla quale è stato eseguito lo script stesso (nei due casi dell'esempio precedente, le cartelle di destinazione saranno `C` e `I` rispettivamente).

1.3. stampare su STDOUT il numero di file copiati.

2. se eseguito come `perl cerca_e_copia.pl PATH_ASSOLUTO` lo script dovrà:

2.1. cercare tutti i file ".pl" che si trovano nella cartella indicata dal path `PATH_ASSOLUTO`.

2.2. eseguire quanto richiesto ai punti 1.2 e 1.3.