Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ____ Col ____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
PROVA SCRITTA DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2021/2022
21 giugno 2022

Esercizio -1: Essere iscritti su AlmaEsami per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione in tutti i fogli prima di svolgere ogni altro esercizio. Scrivere esclusivamente a penna senza abrasioni. E' vietato l'uso delle penne cancellabili, della matita, dei coprenti bianchi per la correzione (bianchetto) e la scrittura in colore rosso (riservato alla correzione).
Il compito e' formato da tre fogli, sei facciate compresa questa. Le soluzioni che si vogliono sottoporre per la correzione devono essere scritte negli spazi bianchi di questi fogli. Non verranno corretti altri supporti.
E' obbligatorio consegnare il compito, e' possibile chiedere che esso non venga valutato scrivendo “NON VALUTARE” in modo ben visibile nella prima facciata.
Per svolgere questo compito occorre solo una penna e un documento di identità valido. La consultazione o anche solo la disponibilità di altro materiale comporterà l'annullamento del compito (verrà automaticamente valutato gravemente insufficiente).

Esercizio c.1: Scrivere il monitor collocamento:
void cercolavoro(char *nome, char *skill)
void char *assumo(char * skill)

Quando un processo chiama la `cercolavoro` si mette in attesa di una richiesta di lavoro e rimane bloccato nel monitor fino a che non è stato assunto. Nella `cercolavoro` viene indicato il nome dell'aspirante lavoratore la sua capacità (skill). Un datore di lavoro con necessità di personale chiama la `assumo` specificando la capacità richiesta. Se c'è in attesa almeno un aspirante lavoratore con quella specifica capacità (uguale valore di skill), il datore di lavoro riceve il nome del nuovo dipendente ed entrambi i processi escono dal monitor. Nel caso non ci siano richieste compatibili il datore di lavoro si blocca nel monitor attendendo un lavoratore con la capacità cercata. Quando arriva il lavoratore che soddisfa le richieste si sbloccano entrambi i processi *lavoratore* e *datore di lavoro.* La `assumo` restituisce in ogni caso il nome del dipendente da assumere.

Esercizio c.2: Un servizio viene fornito in modalità client-server usando message passing asincrono.
Al fine di aumentare l'efficienza si decide di usare molti server e un processo dispatcher in grado di distribuire le richieste agli N server. Quando un processo server è libero riceve dal dispatcher la prossima richiesta da elaborare:
codice di ogni client (tanti!): .....

```
asend(<getpid(), request>, dispatcher)
result = arecv(dispatcher)
```

```
server process[i],  i = 0, ..., N-1:
     request = arecv(dispatcher)
     result = compute(request)
     asend(<getpid(), result>, dispatcher)
```

Scrivere il processo dispatcher. (il dispatcher conosce i pid di tutti i server).

Esercizio g.1: Un sistema monoprocessore usa uno scheduler multilivello a 2 livelli. I processi ad alto livello di priorità vengono gestiti con un algoritmo FIFO mentre quelli a basso livello di priorità usano un algoritmo round-robin con quanto di tempo di 3ms.
Esistono nel sistema due processi periodici ad alto livello di priorità (chiamati h e k) che vengono riattivati ogni 6ms, elaborano per 1ms e terminano. Ci sono inoltre 2 processi a basso livello di priorità (che chiameremo P e Q) che devono fare le seguenti operazioni:
P:: elaborazione 4ms, input-output 2ms, elaborazione 2ms, input-output 1ms, elaborazione 5ms
Q:: elaborazione 5ms, input-output 3ms, elaborazione 2ms, input-output 1ms, elaborazione 4ms
Mostrare il diagramma di Gannt spiegando come è stato calcolato.

Esercizio g.2: rispondere alle seguenti domande (motivando opportunamente le risposte):
a) Come si calcola la lunghezza massima di un file in un FIle System tipo UNIX (bffs, ext2, minix ecc)?
b) Come si fa a risolvere una situazione di deadlock?
c) Perché è meglio la paginazione della compattazione di memoria?
d) Per implementare un servizio di autorizzazione di tipo capability è meglio usare crittografia simmetrica o crittografia a chiave pubblica? perché? E' necessario usare un metodo specifico (crittografia simmetrica o crittografia a chiave pubblica) o il servizio di autorizzazione potrebbe essere implementato con entrambi?