1. Quali sono gli obiettivi generali di un S.O.?

a) Gli obiettivi di un sistema operativo sono:

- La REALIZZAZIONE di una macchina ASTRATTA, che consiste nell'implementare funzionalità di alto livello nascondendo i dettagli del livello più basso;
- Una gestione delle risorse del sistema utilizzando l'HW della macchina in modo efficiente.
- Rendere il sistema di calcolo più semplice all'utente che lo utilizza
- Eseguire i programmini dell'utente e rendere più facile la soluz. dei problemi dell'utente.

b) Si descrivono le COMPONENTI PRINCIPALI di un S.O.

Le componenti di un sistema operativo sono:

- gestione dei file: il S.O. si occupa della CREAZIONE/CANCELLAZIONE dei file e delle directory, dell'allocazione dei file nella memoria secondaria, del salvataggio di dati su supporti non volatili ed il supporto di PRIMITIVE per la gestione di file e directory

- gestione dell' I/O: il S.O. si occupa di fornire un' interfaccia ai gestori dei dispositivi collegati e di fornire i driver per ogni dispositivo esterno collegato al sistema.
I sistemi utilizzati per gestire l'I/O sono: BUFFERING, SPOOLING e CACHING.

- gestione della MEM. PRINCIP.: il S.O. deve tenere traccia di chi e sta utilizzando la MEM PRINCIP e quale parte sta utilizzando;
deve decidere quale processo caricare in memoria quando si libera lo spazio ed infine deve allocare e deallocare spazio in memoria quando gli viene richiesto

- gestione delle MEM. SECONDARIE: il S.O. deve gestire lo spazio delle mem secondarie eventualmente allocando spazio quando necessario. Inoltre deve esserci la schedulazione dei dischi nel caso di più memorie.

- gestione dei PROCESSI: il S.O. deve occuparsi della creaz/cancellaz./sospensione e ripresume dei processi fornendo loro anche dei meccanismi di sincronizz. (evitando i deadlock) e comunicazione (messaggi o con mem. condivisa)

- Sistemi di protezione: il S.O. fornisce dei meccanismi per controllare l'accesso da programmi, processi e utenti sia al S.O. che ai dati salvati.

- networking: deve poter creare un sistema distribuito dove ogni processore ha una memoria propria e tutti i processori sono connessi tramite una rete di comunicazione

- interprete comandi: l'interprete di sistema ha il compito di eseguire un comando, eseguirlo (è figlio del processo padre esegue il comando) e tornare in ascolto in attesa di altri comandi

2) a) Che cosa si intende per processi I/O-Bound e CPU-Bound? Quali effetti si presentano a CPU-Bound?

I processi I/O-Bound sono processi che durante la loro "vita" richiedono molta attività di I/O e poca attività di CPU.
Viceversa i processi CPU-Bound richiedono molta attività di CPU e poca attività di I/O.
Un effetto che si ha dando la precedenza ai processi CPU-BOUND è l'EFFETTO CONVOGLIO in cui non viene eseguito il "parallelismo" perché il primo processo che accede in coda è I/O e consuma una buona "fetta" di CPU bloccando l'esecuzione dei processi I/O in attesa.
Questo effetto può causare un rallentamento dell'interazione tra utente (dispositivi I/O e macchina)

b) Si dicano alcuni esempi di algoritmi che privilegiano I/O Bound.

Un esempio ~~sono~~ è lo scheduling con FEEDBACK, in cui vi sono più code (ciascuna con priorità diversa) e i processi I/O hanno la ~~massima~~ priorità più alta rispetto ai CPU-Bound
Un esempio effettivo è l'algoritmo di scheduling di Linux che favorisce i processi interattivi assegnandoli una priorità dinamica

5.

	C				Max			
	A	B	C	D	A	B	C	D
P_0	2	0	0	1	4	2	1	2
P_1	3	1	2	1	5	2	5	2
P_2	2	1	0	3	2	3	1	6
P_3	1	3	1	2	1	4	2	4
P_4	1	4	3	2	3	6	6	5

A (3, 3, 2, 1)

R = Max - C ⇒

	A	B	C	D	
P_0	2	2	1	1	✓
P_1	2	1	3	1	
P_2	0	2	1	3	✓
P_3	0	1	1	2	✓
P_4	2	2	3	3	✓

a) dimostra che STATO è SICURO

$P_0 \to (5,3,2,2) \to P_3 \to (6,6,3,4) \to P_4 \to (7,10,6,6) \to P_2 \to (9,11,6,9) \to P_1 \to (12,12,8,10)$

b) arriva la richiesta di P_1 per (1,1,0,0), può essere soddisfatta? SI

	C				A	R			
	A	B	C	D	(2,2,2,1)	A	B	C	D
P_0	2	0	0	1		P_0	2	2	1	1	✓
P_1	4	2	2	1		P_1	1	0	3	1	✓
P_2	2	1	0	3		P_2	0	2	1	3	
P_3	1	3	1	2		P_3	0	1	1	2	✓
P_4	1	4	3	2		P_4	2	2	3	3	✓

e cerco una seq sicura

$P_0 \to (4,2,2,2) \to P_3 \to (5,5,3,4) \to P_4 \to (6,9,6,6) \to P_1 \to (10,11,8,7) \to P_2 \to (12,12,8,10)$

Ok può essere soddisfatta → STATO SICURO

c) arriva la richiesta di P_4 per (0,0,2,0), può essere soddisfatta? NO

	C				R			
	A	B	C	D	A	B	C	D
P_0	2	0	0	1	P_0	2	2	1	1
P_1	3	1	2	1	P_1	2	1	3	1
P_2	2	1	0	3	P_2	0	2	1	3
P_3	1	3	1	2	P_3	0	1	1	2
P_4	1	4	5	2	P_4	2	2	1	3

A (3, 3, 0, 1)

e cerco una seq. sicura.

non esiste una seq. sicura in quanto la risorsa C è richiesta da ogni processo ma non ce ne è nessuna disponibile

3. sched. con 3 code A, B, C ; PRIORITÀ CRESCENTE con prelaz. tra code

A : FCFS
B : RR $q = 15$ ms
C : RR $q = 10$ ms

Se un processo consuma il suo quanto, va ~~rimanda~~ in fondo alla coda meno prioritaria

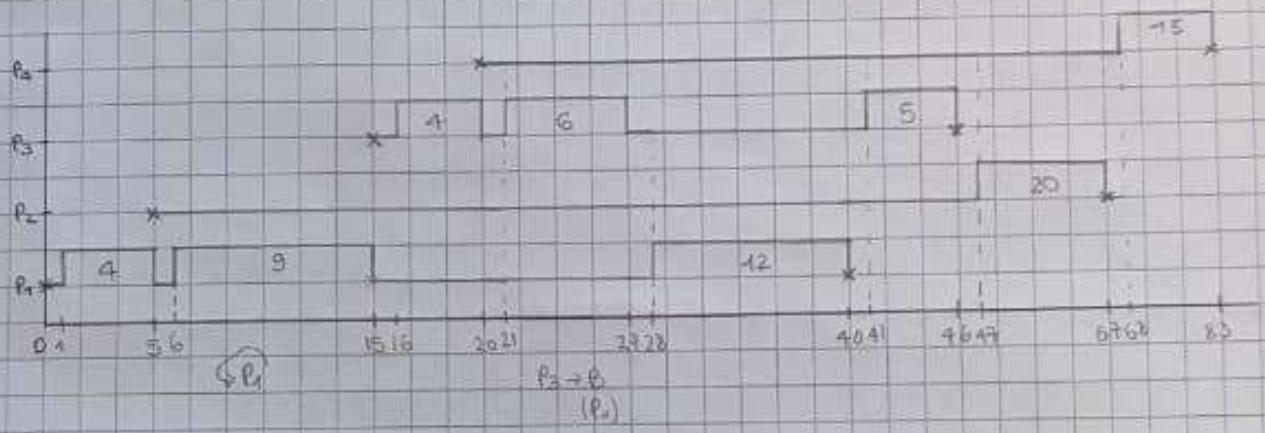
$B \to A$; $C \to B \to A$

	CODA	ARRIVO	BURST
P_1	B	0	25
P_2	A	5	20
P_3	C	15	15
P_4	A	20	15

tempo fat. $k = 1$ ms

se non finisce il quanto, torna in dietro alla coda con quanto pieno
quando arriva un processo in coda a priorità inferiore (che c'è il tempo di cattività) il quanto non si consuma e il processo non torna in coda

traccia il DIAGRAMMA di GANTT



4. I 3 REQUISITI della SEZ. CRITICA sono:

- MUTUA ESCLUSIONE : un processo può entrare nella propria sez. critica soltanto se ha il turno assegnato, o se gli altri processi sono in stato idle (compreso quello che ha il turno assegnato) — ovvero se turn = i

- PROGRESSO : un processo fuori dalla propria sez. critica non può bloccare gli altri in quanto l'unica modifica che viene fatta alla variabile Turn è proprio dentro alla sez. critica.

- ATTESA LIMITATA : quando un processo richiede di entrare nella propria sez. critica deve aspettare un numero finito di volte, nel caso peggiore $n-1$ volte

con l'if((j >= n) && (turn == i || flag[turn] == idle))
controlla se che non ci siano altri processi attivi nella sez. critica ed è il ~~mio~~ mio (i) turno — o chi ha il turno è inidle

↳ se è vero con break esce dal while (true) ed entra nella sez. critica

con turn = j nella sez. critica garantisce il progresso