1. **[2 boda] Korištenjem raspoređivanja prema najmanjoj labavosti (engl. *least laxity first*) te redu prispijeća kao sekundarnim kada prvi ne daje jedan zadatak, grafički prikazati raspoređivanje sustava zadataka na dvoprocesorskom sustavu u intervalu [0,30].**

$\mathcal{T}_1$: $T_1 = 10$ **ms**, $C_1 = 5$ **ms**
$\mathcal{T}_2$: $T_2 = 15$ **ms**, $C_2 = 7$ **ms**
$\mathcal{T}_3$: $T_3 = 20$ **ms**, $C_3 = 10$ **ms**
$\mathcal{T}_4$: $T_4 = 30$ **ms**, $C_4 = 15$ **ms**

Rj. (sve u [ms], korak = 1 ms, <> označava odabrane zadatke)

t	c_1, l_1	c_2, l_2	c_3, l_3	c_4, l_4
0	5, <5>	7, <8>	10, 10	15, 15
1	4, <5>	6, <8>	10, 9	15, 14
2	3, <5>	5, <8>	10, 8	15, 13
3	2, <5>	4, 8	10, <7>	15, 12
4	1, <5>	4, <7>	9, 7	15, 11
5	G	3, <7>	9, <6>	15, 10
6	–	2, <7>	8, <6>	15, 9
7	–	1, <7>	7, <6>	15, 8
8	–	G	6, <6>	15, <7>
9	–	–	5, <6>	14, <7>
10	5, <5>	–	4, <6>	13, 7
11	4, <5>	–	3, <6>	13, 6
12	3, <5>	–	2, 6	13, <5>
13	2, 5	–	2, <5>	12, <5>
14	2, <4>	–	1, <5>	11, 5
15	1, <4>	7, 8	G	11, <5>
16	G	7, <7>	–	10, <5>
17	–	6, <7>	–	9, <5>
18	–	5, <7>	–	8, <5>
19	–	4, <7>	–	7, <5>
20	5, <5>	3, 7	10, 10	6, <5>
21	4, <5>	3, 6	10, 9	5, <5>
22	3, 5	3, <5>	10, 8	4, <5>
23	3, <4>	2, 5	10, 7	3, <5>
24	2, <4>	2, <4>	10, 6	2, 5
25	1, 4	1, <4>	10, 5	2, <4>
26	1, <3>	G	10, 4	1, <4>
27	G	–	10, <3>	G
28	–	–	9, <3>	–
29	–	–	8, <3>	–
30	5, <5>	7, 8	7, <3>	15, 15

2. **[2 boda] PID regulator zadan je parametrima $K_P = 10$, $K_I = 3$, $K_D = -2$ te korakom integracije $T = 1$. Ako se reakcija okoline može simulirati formulom $y_{k+1} = y_k + 0,5 \cdot r_k^2$ napraviti tri koraka integracije. Početno stanje sustava je $y_0 = 10$, a željeno stanje $y_P = 20$.**

   Rj.

i	y_i	e_i	I_i	D_i	r_i
0	10	10	10	10	110
1	6060	-6040	10	-6050	-48270
2	1165002510	-1165002490	10	-1164996450	-9320031970

3. **[1 bod] Opisati `SCHED_FIFO` način raspoređivanja.**

   Rj.
   - uvijek se odabire dretva najvećeg prioriteta (na višeprocesorskim sustavima N dretvi najveća prioriteta)
   - kada ima više dretvi istog najvećeg prioriteta, odabire se ona koja je prije došla u red pripravnih

4. **[1 bod] Navesti sučelja operacijskih sustava za postavljanje parametara raspoređivanja dretvi (POSIX nazive ili opisni naziv, uz opis što rade).**

   Rj.
   - `postavi_način_raspoređivanja`
   - `postavi_prioritet`
   - `postavi_način_raspoređivanja_i_prioritet`
   - gornje, ali kao parametar za stvaranje nove dretve
   - postavljanje protokola nasljeđivanja prioriteta ili protokola stropnog prioriteta za neko sredstvo sinkronizacije (npr. monitor)

5. **[1 bod] Raspoređivanje nekritičnih dretvi modelira se višerazinskim raspoređivanjem s povratnom vezom (engl. *multilevel feedback queue – MFQ*). Opisati taj model.**

   Rj.
   - koristi se više redova za dretve
   - uvijek se uzima prva dretva iz najvišeg reda
   - nova dretva ide u najviši red (na kraj tog reda)
   - kad se dretvi daje procesor daje joj se na kvant vremena
   - ako dretva nije gotova u kvant vremena, zaustavlja se i miče u red ispod gdje je prije bila (smanjuje joj se prioritet za 1)
   - ako je dretve gotova prije isteka kvanta, ona nestaje iz sustava
   - ako se dretva blokirala pri izvođenju, po povratku se vraća na isto mjesto (isti red) ili čak i viši
   - ako dretva dođe do dna (najmanji prioritet) tu ostaje i dijeli procesorsko vrijeme s ostalim dretvama u tom redu (za te se dretve koristi raspoređivanje podjelom vremena)

6. **[1 bod] Navesti sučelja operacijskih sustava za korištenje vremena (dohvat, odgoda, alarmi, POSIX nazive ili opisni naziv, uz opis što rade).**

Rj.
- dovati_sat (id_sata, t) – dovat sata "id_sata"
- postavi_sat (id_sata, t) – postavi novu vrijednost sata "id_sata"
- odgodi (id_sata, t) – odgodi dretvu za "t" po satu "id_sata"
- odgodi_do (id_sata, t) – odgodi dretvu do trenutka "t" po satu "id_sata"
- stvori_alarm (akcija, alarm) – stvori alarm koji radi "akciju" i vrati opisnik u "alarm"
- postavi_alarm (alarm, odgoda, periodička_odgoda) – pokreni alarm

7. **[1 bod] Osim uobičajenih sinkronizacijskih potencijalno blokirajućih funkcija tipa `Cekaj_Nesto`, za SRSV-e su ponekad potrebna i neka proširenja. Koja?**

Rj.
- Probaj_Čekati (try_wait) – ne blokiraj dretvu već vrati grešku u slučaju da se sredstvo ne može zauzeti
- Čekaj_Kratko (timed_wait) – blokiraj ako treba, ali ne beskonačno već najviše zadano vrijeme

8. **[2 boda] U nekom sustavu javljaju se dretve: D_1 u $t_1 = 0$ ms, D_2 u $t_2 = 3$ ms te D_3 u $t_3 = 6$ ms. Svaka dretva sastoji se od tri dijela posla: A, B i C. Izvođenje A dijela traje 2 ms, B dijela 3 ms te C dijela 2 ms. Prije izvođenja B dijela dretve trebaju zauzeti semafore: D_1 semafor S_1, D_2 semafor S_2, D_3 semafor S_1. Dretva D_3 ima najveći prioritet, dok D_1 ima najmanji. Ukoliko se koristi protokol nasljeđivanja prioriteta pokazati izvođenje zadanih dretvi, tj. što procesor radi u pojedinom trenutku, dok sve dretve ne završe sa svojim poslovima.**

Rj.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
D	1_A	1_A	1_B	2_A	2_A	2_B	3_A	3_A	1_B	1_B	3_B	3_B	3_B	3_C	3_C	2_B	2_B	2_C	2_C	1_C	1_C	1_C	–

9. **[1 bod] Između očitanja neke ulazne naprave treba proći između 20 i 50 sabirničkih ciklusa. Kako to ostvariti ako se može programirati u asembleru, te kako ako se mora koristiti C?**

   Rj.
   "Brojanje" sabirničkih ciklusa u modernim arhitekturama je vrlo problematičan posao. Razlog tome jest korištenje priručnih međuspremnika koji rade i do nekoliko puta brže od glavne sabirnice sustava.
   Kod jednostavnog mikrokontrolera bez priručnog spremnika (cache-a) bi navedenu odgodu mogli ostvariti jednostavnom praznom petljom:

   ```
   za i=0 do 20/X+1 radi
      ništa;
   ```

   uz pretpostavku da će instrukcije dohvaćati iz spremnika i da će mu za jednu iteraciju trebati X instrukcija
   U C-u treba paziti da prevoditelj (kompiler) ne izbaci praznu petlju u postupku optimizacije. Za to se može koristiti memorijska barijera ili nekakav nepotreban posao za koji prevoditelj ne može znati je li koristan ili nije.
   U složenijim sustavima, koji imaju procesor s privremenim međuspremnikom (cache-om), navedeno se može napraviti na nekoliko načina. Jedan od njih jest korištenjem poznata odnosa brzine pristupa glavnom spremniku i privremenom spremniku, tj. proračunom koliko treba iteracija izvesti da bi se dobilo trajanje odgovarajućeg broja sabirničkih ciklusa. U C-u, kao i u prethodnom slučaju, treba paziti na optimiranje!

10. **[1 bod] Za dodjelu jedinstvenih identifikacijskih brojeva koristi se funkcija:**
    `int id ()static int broj = 0; return broj++;.`
    Koje sve nedostatke ima navedena funkcija?

    Rj.
    Prvi problem jest u problematičnom korištenju od strane više dretvi. Ako se operacija `broj++` ne obavi atomarno, moguće je da bi dvije dretve dobile isti broj.
    Rješenje za ovaj problem bi bilo u korištenju atomarnih instrukcija ili zaključavanje koda koji povećava varijablu.

    Drugi problem jesu granice tipa podatka `int`. Naime, nakon što se dođe do najveće vrijednosti (uobičajeno $2^{31}-1$) iduća vrijednost je negativna (-2^{31}).
    Rješenje ovog problema bi mogli programski riješiti da se pazi da povećanje od najveće pozitivne vrijednosti ide na nulu (ili 1), a ne negativno. Ili umjesto `int` koristiti `unsigned int` tip podataka ili slično (`unsigned long`).

11. **[1 bod] Koju podršku nude operacijski sustavi za upravljanje napravama mehanizmom prekida? Što ako bi obrada mogla duže potrajati?**

    Rj.
    Za neki prekid preko operacijskog sustava može se registrirati funkcija za obradu prekida (iako u složenijim operacijskim sustavima se to ne radi iz korisničkih programa nekog iz jezgre, kada se u sustavu pojavi neka nova naprava).
    U slučaju kada bi obrada prekida mogla potrajati, neki operacijski sustavi nude mogućnost da duži dio obrade prekida ide naknadno: najprije se obavi osnovni dio, a kasnije, s dozvoljenim prekidanjem i duži dio (u nekoj dretvi koja obrađuje te dijelove). Primjeri na Windows-ima: ISR+IST, Linux: Top half + Bottom half.

12. **[1 bod] Koji su mogući problemi primjene upravljanja spremnikom straničenjem u SRSV-ima?**

Rj.
Problem jest što bi promašaj mogao odgoditi izvođenje dretve u, za njeno upravljanje kritičnom trenutku. Zato se straničenje ili ne koristi ili se zaključavaju kritični procesi u radni spremnik da se promašaji ne bi dogodili.

13. **[2 boda] Opisati osnovna načela i svojstva međudretvene komunikacije korištenjem mehanizama zajedničkog spremnika, reda poruka, cjevovoda i signala. Kada koristiti pojedine mehanizme?**

Rj.
Zajednički spremnik: dretve unutar istog procesa izravno pristupaju dijeljenim podacima; najbrži način komunikacije, ali vrlo često zahtijeva sinkronizaciju! Dretve različitih procesa trebaju od operacijskog sustava najprije pribaviti segment spremnika koji će njihovi procesi dijeliti (moći pristupiti iz oba procesa).
Red poruka: u red jedne dretve stavljaju poruke a druge ih uzimaju. Imaju smisla kada su poruke male, obzirom da svaka poruka ima zaglavlje, svaka je poruka podatak "za sebe".
Cjevovod: u jednu stranu cjevovoda se stavljaju podaci, a s druge se čitaju. Imaju smisla kada treba više podataka prenijeti između dviju strana (npr. datoteke).
Signali: ne prenose podatke nego dojavljuju događaje na koje je moguće "trenutno" reagirati.

14. **[1 bod] Koji su razlozi nepouzdanosti (loših vremenskih svojstava) mreže koja se temelji na Internetu i njegovim protokolima?**

Rj.
Jedan od najvećih problema jest složenost, odnosno, slojevitost protokola. Svaki protokol mora obaviti svoj dio posla prije predavanja nižem sloju ili slanju u viši sloj. Nadalje, problem jest u korištenju međuspremnika – iako oni u inače popravljaju učinkovitost sustava (npr. broj prenesenih podataka u jedinici vremena) oni dodatno generiraju kašnjenja (podaci se šalju tek kad se skupi dovoljno podataka). Zbog gornjih svojstava protokola, svi uređaji koji sudjeluju u komunikaciji između dva udaljena računala unose dodatna kašnjenja, koja nije moguće postaviti u neke ograde obzirom da svaki takav uređaj može u različitim trenucima biti različito opterećen ostalim prometom (ili čak i "napadima").

15. **[2 boda] U nekom sustavu treba uskladiti sat klijentskog računala s poslužiteljem. Za tu svrhu klijent šalje poruku u koju stavlja vrijeme slanja ($^kt_1 = 10$ ms). Po primitku poslužitelj u poruku stavlja vrijeme primitka ($^pt_2 = 30$ ms) te šalje odgovor (u $^pt_3 = 60$ ms). Po primitku odgovora ($^kt_4 = 50$ ms) klijent treba izračunati razliku u satovima i dodati tu vrijednost na svoj sat. Koliko milisekundi treba klijent dodati na svoj sat?**

Rj.
Obzirom da drukčije nije naglašeno u zadatku pretpostavlja se da je veza između klijenta i poslužitelja simetrična: jednako traju slanje i primanje. Uz tu pretpostavku može se odrediti zajednički trenutak kod klijenta i poslužitelja, a to je sredina intervala:

$$^kt = (^kt_1 + {}^kt_4)/2 = (10 + 50)/2 = 30$$
$$^pt = (^pt_2 + {}^pt_3)/2 = (30 + 60)/2 = 45$$
$$c = {}^pt - {}^kt = 45 - 30 = 15\,\text{ms}$$