Zadaci i neka moguća rješenja.

1. [1 bod] Za izgradnju programske potpore koja bi trebala jako dugo biti u uporabi treba odabrati između nekoliko modela procesa izgradnje: evolucijskog, inkrementalnog, komponentno usmjerenog i iterativnog. Koji je model najprikladniji ukoliko ima dovoljno vremena za izradu i sustav treba isporučiti s punom funkcionalnošću? Obrazložiti.

   Osim evolucijskog, koji ne izgrađuje dobru arhitekturu (skupo održavanje i nadogradnje), svi ostali (uz dobro objašnjenje) mogu biti dobar izbor. Možda je iterativni u maloj prednosti; ali i ostali (inkrementalni i komponentni) također mogu ujedno biti i iterativni...

2. [1 bod] Može li se preko varijable okoline razmjenjivati podatke između dva procesa na istom računalu? Npr. da preko varijable okoline VAR1 prvi proces šalje podatke drugom, a preko varijable VAR2 drugi šalje prvom. Obrazložiti.

   NE! Promjena varijable okoline vidljiva je samo u tom procesu. Eventualni procesi koji nastaju iz njega nasljediti će tu varijablu, ali bilo kakva naknadna promjena neće biti vidljiva u oba procesa.

3. [1 bod] Što je sve potrebno napraviti da bi dva procesa mogla razmjenjivati podatke preko reda poruka?

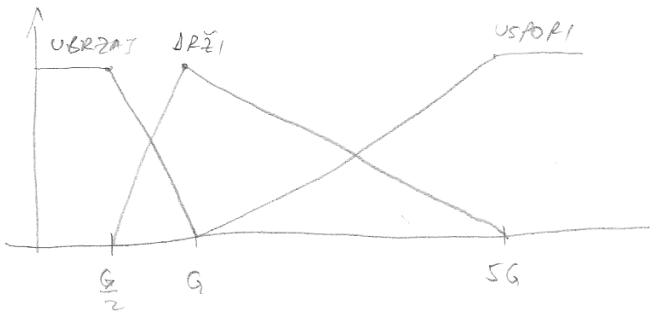
   Oba procesa moraj znati ime (ili ključ) reda kojeg žele koristiti (u lab.vj. se za to koristila varijabla okoline).
   Procesi potom trebaju otvoriti red poruka i ako ne postoji napraviti ga.
   Komunikacija je slanje i primanje (jedna strana šalje druga prima; ako se želi dvosmjerna komunikacija potrebno je otvoriti drugi red za drugi smjer komunikacije – inače bi proces koji je poslao poruku istu mogao i pročitati).

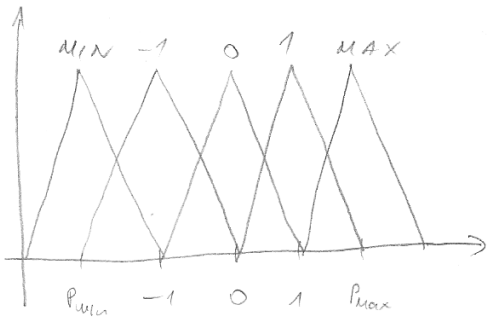
4. [1 bod] Što je sve potrebno napraviti prije korištenja zajedničkog spremnika (engl. *shared memory*), npr. prije upisivanja podataka u njega?

   1. Otvoriti/stvoriti segment zajednička spremnika (shm_open).
   2. Postaviti mu veličinu (ftruncate).
   3. Mapirati ga u adresni prostor procesa (mmap).

5. [2 boda] Upravljanje potisnicima neke rakete treba ostvariti korištenjem neizrazite logike. Pretpostaviti da raketa (u intervalu za koje treba ostvariti upravljanje) ima postavljeno željeno ubrzanje G. Kada je ubrzanje manje, snagu potisnika treba povećavati, a kada je ubrzanje veće snagu smanjivati. Ukoliko je ubrzanje manje od $G/2$ treba koristiti maksimalnu snagu P_{MAX}. Od $G/2$ do G promjena snage treba padati do nule (kada je ubrzanje upravo jednako G ostaviti trenutnu snagu). Od G do $5 \cdot G$ snagu smanjivati do minimalne vrijednosti snage P_{MIN} u točki $5G$. Ukoliko je ubrzanje veće od $5G$ snaga potisnika treba biti P_{MIN}. Skicirati ostvarenje upravljanja. Ulaz je trenutno ubrzanje, a izlaz promjena snage s vrijednostima: P_{MIN}, -1, 0, 1, P_{MAX} (uz granične vrijednosti se snaga automatski postavlja na zadano – tada to nije "povećanje/smanjenje" snage).

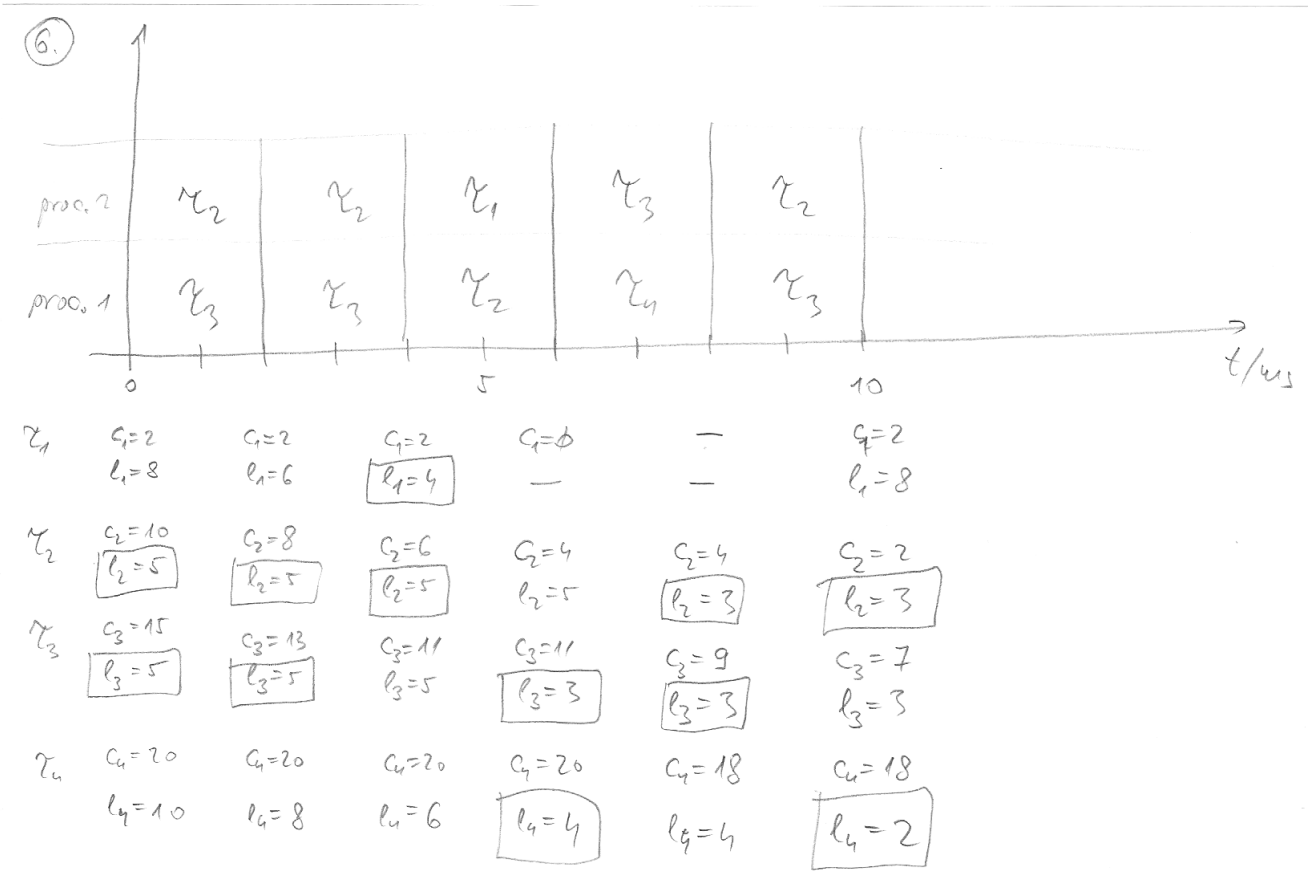
(5)





1. ako je $G \in$ UBRZAJ tada $\Delta P \in$ MAX
2. ako je $G \in$ UBRZAJ i $G \in$ DRŽI tada $\Delta P \in 1$
3. ako je $G \in$ DRŽI tada $\Delta t \in 0$
4. ako je $G \in$ DRŽI i $G \in$ USPORI tada $\Delta P \in -1$
5. ako je $G \in$ USPORI tada $\Delta P \in$ MIN

6. [1 bod] Sustav zadataka sastoji se od četiri periodička zadatka: prvi se javlja svakih 10 ms i treba 2 ms procesorskog vremena, drugi svakih 15 ms i treba 10 ms, treći svakih 20 ms i treba 15 ms te četvrti svakih 30 ms i treba 20 ms. Prikazati raspoređivanje navedena sustava u kritičnom slučaju korištenjem raspoređivanja prema najmanjoj labavosti do t = 10 ms na dvoprocesorskom sustavu. Korak s kojim se prekidaju dulje obrade i u kojem se nanovo poziva raspoređivač je 2 ms. Sekundarni kriterij (kada prvi nije dovoljan da jednoznačno odabere 2 zadatka) je mjera ponavljanja (RMPA).



7. [2 boda] U nekom sustavu početno se nalazi jedna dretva D_1 prioriteta $p_1 = 3$ koja treba još $c_1 = 15$ ms procesorskog vremena. U 5. ms javlja se dretva D_2 prioriteta $p_2 = 5$ i potrebom od $c_2 = 10$ ms. U 10. ms javlja se dretva D_3 prioriteta $p_3 = 5$ i potrebom do $c_3 = 5$ ms. Dretve D_1 i D_2 raspoređuje se prema SCHED_RR dok dretva D_3 prema SCHED_FIFO. Prikazati rad sustava dok sve navedene dretve ne obave svoje poslove. Kvant vremena za SCHED_RR je $T_q = 1$ ms.

```
Oznake: 1=>D1, 2=>D2, 3=>D3

PROC: 11111222223333322222111111111

ili (isto točno kada nova dretva odmah dobiva kvant)

PROC: 111112222223333322222111111111
```

8. [1 bod] Neki upravljački program treba zaštititi nadzornim alarmom čije se brojilo nalazi na adresi 0x447138, i koje otkucava s 1 MHz. Program s maksimalno dozvoljenim trajanjima dijelova zadan je prema:

```
ponavljaj {
    očitaj_senzore();  //najdulje 100 mikrosekundi
    obavi_proračun();  //najdulje 500 mikrosekundi
    pošalji_naredbe(); //najdulje 50 mikrosekundi
}
```

Proširiti program tako da se u slučaju prekoračenja zadanih trajanja sustav resetira.

```
int *na = (int *) 0x447138;
ponavljaj {
    *na = 100;
    očitaj_senzore();  //najdulje 100 mikrosekundi
    *na = 500;
    obavi_proračun();  //najdulje 500 mikrosekundi
    *na = 50;
    pošalji_naredbe(); //najdulje 50 mikrosekundi
}
```

9. [2 boda] Dretva D_1 upravlja s aktivnostima $A_1 - A_5$ (pozivima `akt_1(i)`, gdje je `i` oznaka aktivnosti). Upravljanje aktivnošću A_2 povremeno radi i druga dretva D_2 (pozivom `akt_2(2)`) te se to upravljanje ne smije izvoditi istovremeno. Sve su aktivnosti podjednako bitne i zahtjev je da se te aktivnosti što češće provjeravaju od strane dretve D_1 (pozivaju navedene funkcije). Ponašanje dretve D_2 se ne može predvidjeti (koliko će često ona pozivati `akt_2(2)`), ali joj se može taj poziv omotati dodatnim kodom. Ostvariti prikladno rješenje i opisati kako ono zadovoljava navedeni zahtjev.

```
dretvi D2 opoziv akt_2(2) omotati semaforom:
    ...
    ČekajSemafor(KO)
    akt_2(2)
    PostaviSemafor(KO)
    ...

dretva D1:
    ponavljaj {
        akt_1(1)
        ako je ProbajČekatiSemafor(KO) == 0 tada
            akt_1(2)
            PostaviSemafor(KO)
        //inače preskoči upravljanje 2. aktivnosti u ovoj
        //iteraciji jer to trenutno radi druga dretva
        akt_1(3)
        akt_1(4)
        akt_1(5)
    }
```

10. [2 boda] U nekom sustavu nalaze se dretve $D_1 - D_4$ s prioritetima $p_1 = 20$, $p_2 = 15$, $p_3 = 10$, $p_4 = 5$. Poznato je da će dretva D_1 u svom radu trebati semafore S_1 i S_2, dretva D_2 semafore S_2 i S_3, dretva D_3 semafore S_3 i S_4 te dretva D_4 semafore S_1 i S_3. Početne vrijednosti svih semafora su 1. Ukoliko se koristi izvorni protokol stropnog prioriteta opisati zbivanja u sustavu za slijedeći scenarij:
a) u t = 0 se javlja D_4 i treba S_3
b) u t = 5 se javlja D_2 i treba S_2
c) u t = 10 se javlja D_1 i treba S_1
d) u t = 15 se javlja D_3 i treba S_3
Pretpostaviti da svaka dretva nakon što zauzme semafor isti i vrati nakon 20 ms korištenja, osim dretve D_1 koja nakon 10 ms treba i semafor S_2, te nakon još 10 ms rada s oba semafora, oba i otpušta.

10.

D	p	S
D_1	20	S_1, S_2
D_2	15	S_2, S_3
D_3	10	S_3, S_4
D_4	5	S_1, S_3

S	SP
S_1	20
S_2	20
S_3	15
S_4	10

t=0 D_4 zauzima S_3

t=5 D_2 se blokira ($P_2 > SP(S^* = S_3) = 15$ nije zad.) $\Rightarrow p(D_4) = p(D_2) = 15$

t=10 D_1 zauzima S_1 (D_4 je do tada obradila 10 ms)

t=15 D_3 se blokira ($p(D_4) = 15 > p(D_2)$ pa D_4 u pozitivnom prioritetu od D_3)

(pretpostavimo jednoprocesorski sustav)

t=20 D_1 zauzima S_2 ($S^* = S_1$ koji drži brdra D_1!)

t=30 D_1 otpušta S_1 i S_2 (i završava), D_4 nastavlja s radom

t=40 D_4 otpušta S_3 $\Rightarrow$ vraća joj se prio na 5, D_2 nastavlja s radom - zauzima S_2

t=60 D_2 otpušta S_2, završava, D_3 zauzima S_3

t=80 D_3 otpušta S_3, završava

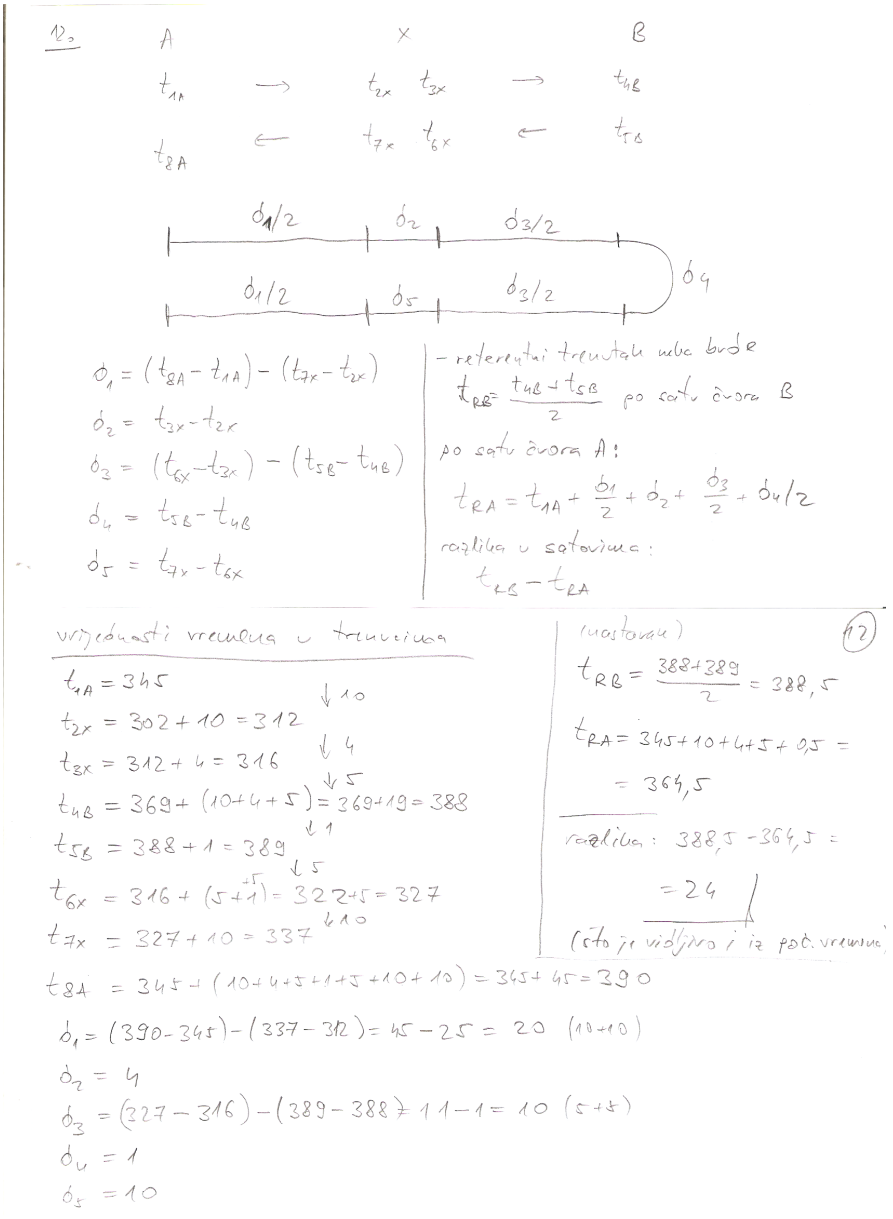
11. [1 bod] Zašto TCP nije dobar protokol za korištenje u SRSV-ima?

TCP nudi pouzdan prijenos: sve što se pošalje stići će do odredišta (osim ako se veza prekine), sam radi retransmisije. Nije moguće utjecati na to - reći da ne radi retransmisiju, da irelevantne podatke ne pokušava više slati već da šalje nove.
Taj nedeterminizam nije dobar za SRSV gdje je potrebna potpuna kontrola biti u upravljačkom programu.

12. [2 boda] Čvor A koristi čvor X za komunikaciju s čvorom B. Međutim, čvor X je opterećen i drugim poslovima te se vrijeme od prihvata poruke od čvora A ili B do njegovog prosljeđivanja na drugi čvor može znatno razlikovati u različitim trenucima (čak i bliskim). Poznato je da prijenos od A do X traje jednako kao i od X do A, te prijenos od X do B isto kao i od B do X. Uz navedene pretpostavke, osmisliti algoritam (koji bi se mogao ugraditi u sva tri čvora) pomoću kojeg bi čvor A mogao uskladiti svoj sat s čvorom B. Prikazati algoritam na primjeru gdje su početne vrijednosti satova $t_A = 345$, $t_X = 302$ te $t_B = 369$, put od A do X traje 10 jedinica; X nakon 4 jedinice šalje poruku (izvornu ili promjenjenu, prema algoritmu) dalje; put od X do B traje 5 jedinica; B šalje odgovor za 1 jedinicu; put od B do X traje 5 jedinica, od prihvata poruke od B do slanja poruke čvoru A protekne 10 jedinica; put od X do A traje 10 jedinica.

Najjednostavnije rješenje je:
1. uskladiti sat čvora X s B-om (ili bar virtualni sat X-a)
2. uskladiti sat čvora A s X-om
Formule za navedeno su u skripti.

Druga moguća rješenja uključuju 'crtanje' grafa, i izgradnju funkcija. Jedan takav primjer je u nastavku.



13. [1 bod] U nekom programu nalazi se dio koda:

```
unsigned long x, y, z, w;
... //dio koda u kojem se koriste i mijenjaju varijable y, z i w
x = y * z / w;
...
```

Navesti moguć/e problem/e i kako ga/ih riješiti, uz pretpostavku da je `w` različit od nule.

```
Umnožak y*z može prelaziti granice tipa unsigned long.
Moguća rješenja:
- (privremeno ili trajno) koristiti tip podatka kod kojeg to neće
biti problem:
  * unsigned long long (ako je na raspolaganju)
  * double (ako je na raspolaganju)
  * npr.
  double tmp = y;
  y = y * z / w;
  x = (unsigned long) y;
- promijeniti algoritam obzirom na očekivane vrijednosti y, z i w
(npr. rastaviti brojeve na 16 bitovne elemente i preraditi formulu
)
```

14. [1 bod] Opisati uobičajene **mogućnosti** operacijskih sustava za SRSV koje su osmišljene da bi se prekid većeg prioriteta što prije prihvatio i obradio.

```
Mogućnosti:
1. Obrada prekida prema prioritetima (niže obrade se prekidaju
onima većeg prioriteta).
2. Obrada prekida se dijeli na dva dijela: prvi (vrlo kratak) se
obavlja odmah, a drugi naknadno, prema prioritetu.
```

15. [1 bod] Neka struktura podataka koristi se iz nekoliko dretvi, ali se u nju treba našto upisati i nakon prekida neke naprave (zbog tog prekida). Struktura je zaštićena semaforom. Navesti moguće probleme ‘naivnog’ rješenja te predložiti bolje rješenje.

```
Naivno rješenje bi koristilo ČekajSemafor i u obradi prekida. Među
tim, to se ne može - obrada prekida se ne smije blokirati.
Ni PokušajČekati nije rješenje, jer što ako ne uspije? I dalje
treba poslati podatke.
Jedno od rješenja jest stvoriti novu dretvu koja će čekati na
takve podatke i ona tada može pozvati ČekajSemafor. Toj dretvi se
iz prekida podaci mogu poslati preko reda poruka, cjevovodom ili
preko zajedničkih spremnika.
(Ili umjesto nove dretve, nekoj od postojećih poslati signal, u č
ijoj će obradi ta dretva napraviti navedeno - ali ta dretva ne
smije koristiti tu istu strukturu u normalnom radu.)
```