Završni ispit iz predmeta: ***Sustavi za rad u stvarnom vremenu*** 29. 1. 2020.

Pisati čitko – nečitak odgovor ne donosi bodove.

1. Neki sustav upravljanja sastoji se od četiri periodička zadatka $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ i $\mathcal{T}_4$ s periodama: $T_1 = 10$, $T_2 = 15$, $T_3 = 25$ i $T_4 = 40$ jedinica vremena. Zadaci svoje poslove obavljaju pozivanjem odgovarajuće funkcije $p_x()$ ($p_1()$ za $\mathcal{T}_1$, ...).

   a) [2 boda] Ostvariti upravljanje navedenim sustavom ukoliko na raspolaganju stoji jedino funkcija za dohvat vremena $dohvati\_vrijeme()$. Minimizirati greške koje nastaju zbog nepoznatog i promjenjivog trajanja izvođenja zadataka.

   b) [2 bod] Proizvoljnim postupkom provjeriti je li sustav zadataka rasporediv na jednoprocesorskom sustavu postupkom mjere ponavljanja uz pretpostavku trajanja zadataka $c_1 = 3$, $c_2 = 5$, $c_3 = 4$ i $c_4 = 6$.

   c) [2 boda] Kolika bi bila zalihost računalne snage u prvih 15 jedinica vremena, ako bi se sustav zadataka izvodio na dvoprocesorskom računalu uz ista trajanja zadataka navedena pod b).

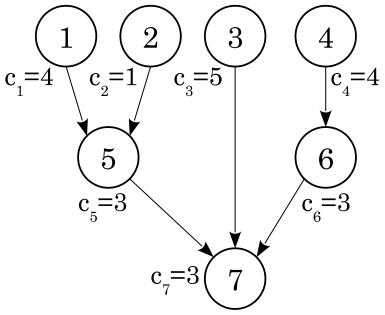
```
a)
T = {10, 15, 25, 40}
t = {0, 0, 0, 0}
p = {p1, p2, p3, p4}

ponavljaj
    za i=1 do 4 radi
        ako je t[i] <= dohvati_vrijeme() tada
            p[i]()
            t[i] += T[i]


b)
simulacija
 1 1 1 2 2 2 2 2 3 3 1 1 1 3 3 2 2 2 2 2 1 1 1 4 4 3 3 3 3 4 1 1 1 2 2 2 2 2 4 4
0-1-2-3-4-5-6-7-8-9-0-1-2-3-4-5-6-7-8-9-0-1-2-3-4-5-6-7-8-9-0-1-2-3-4-5-6-7-8-9-0
1                   1         2         1         3         1                   1
2                                                           2                   4
3                                                                               X4
4                                                                 zad4 ne stigne


c)
u prvih 15 jed. vremena moraju biti gotovi zad1 i zad2 (treba 3 + 5)
ostali imaju veću labavost od 15 i ne trebaju niti započeti!
F(m=2, t=15) = m*15 - (3 + 5) = 30-8 = 22
```

2. [2 boda] Prikazati postupak općeg raspoređivanja nad prikazanim sustavom zadataka uz korištenje dvoprocesorskog računala. Dovoljno je za svaki interval u kojem se zadaci izvode napisati njihove udjele u procesorskom vremenu ("alfe").




```
t1 = 0: s1={z1,z2,z5}, z3, s2={z4,z6}, aktivni: z1,z2,z3,z4
-------
c_s1=8
c_z3=5
c_s2=7
Dt = (8+5+7)/2 = 10
alfa_s1 = 0,8
alfa_z3 = 0,5
alfa_s2 = 0,7
alfa_z1 = alfa_s1 * (c1/(c1+c2)) = 0,8 * 4/5 = 0,64
alfa_z2 = alfa_s1 * (c2/(c1+c2)) = 0,8 * 1/5 = 0,16
t_z1 = t1 + c1 / alfa_z1 = 0 + 4 / 0,64 = 1 / 0,16 = 6,25
t_z2 = t1 + c2 / alfa_z2 = 0 + 1 / 0,16 = 6,25 (očekivano!)
t_z2 = t1 + c3 / alfa_z3 = 0 + 5 / 0,5 = 10
t_z4 = t1 + c4 / alfa_s2 = 0 + 4 / 0,7 = 5,71 => prvi je gotov

t2 = 5,71: aktivni: z1,z2,z3,z6
-------
t_z6 = t2 + c6 / alfa_s2 = 5,71 + 3 / 0,7 = 4 / 0,7 + 3 / 0,7 = 10
z1 i z2 završavaju u t=6,25

t3 = 6,25: aktivni: z5,z3,z6
-------
t_z5 = t3 + c5 / alfa_s1 = 1/0,16 + 3/0,8 = 100/16 + 30/8 = 160/16 = 10

t4 = 10: aktivni z7
-------

t5= 13: sve gotovo
-------
```

3. [2 boda] U nekom jednoprocesorskom sustavu stvaraju se dretve:
   – u t=0 D1 s prioritetom 5 i načinom raspoređivanja SCHED_RR,
   – u t=1 D2 s prioritetom 5 i načinom raspoređivanja SCHED_FIFO,
   – u t=3 D3 s prioritetom 8 i načinom raspoređivanja SCHED_RR, te
   – u t=6 D4 s prioritetom 5 i načinom raspoređivanja SCHED_RR.
   Kvant vremena $t_q = 2$ jedinice vremena. Svaka dretva treba 4 jedinice procesorskog vremena. Prikazati rad sustava dok sve dretve ne završe s radom.

```
simulacija
              |4|
      | 1 | 1 |  4  |
    |1| 2 | 2 |  1  | 4 |
| 1 |2| 3 | 3 |  2  | 1 | 4 | 4 |
0-1-2-3-4-5-6-7-8-9-0-1-2-3-4-5-6-7-8-9-0
1 2   3     4
              3     2   1       4
```

4. [2 boda] Neka dretva treba svoj posao obavljati svakih 100 ms. Koji su SVE nedostatci prikazanog koda kojim se navedeno želi ostvariti, te kako ih otkloniti?

```
void *dretva (void *p) {
    struct timespec t;
    t.tv_sec = 0; t.tv_nsec = 100000;
    while (1) {
        posao_dretve(p);
        clock_nanosleep (CLOCK_REALTIME, 0, &t, NULL);
    }
}
```

```
1. krivo vrijeme odgode je zadano
   + treba dodati 3 nule na tv_nsec (inače je 100 mikrosekundi)
2. CLOCK_REALTIME nije dobar jer se može promijeniti sat
   + treba koristiti CLOCK_MONOTONIC
3. odgoda ne uzima u obzir trajanje posao_dretve(p)
   + treba koristiti odgodu do (dodati zastavicu TIMER_ABSTIME)
     početno uzeti vrijeme i na njega dodati 100 ms
     nakon posla odspavati do tog vremena
     povećavati t za 100 ms u svakoj iteraciji
4. signali mogu raditi probleme:
   + odgodu (prema 3.) vrtiti u petlji: kada vrati -1 ponovno ju pozvati
5. funkcija treba vratiti kazaljku prije izlaska
   (iako se to nikada neće dogoditi zbog beskonačne petlje)
   + npr. return NULL; ili return p;
```

5. [2 boda] Problem pet filozofa riješen je prikazanim kodom sa semaforima uz korištenje izvorna protokola stropnog prioriteta ("složenijeg"). Sve dretve imaju jednaki prioritet=10.

```
filozof (i)
    ponavljaj
        razmišljaj
        ČekajSEM (i)
        ČekajSEM ((i+1) mod 5)
        jedi (traje 10 jedinica vremena)
        PostaviSEM ((i+1) mod 5)
        PostaviSEM (i)
```

Pokazati rad sustava za sljedeće scenarije:

a) u t=0 filozof 1 prvi hoće jesti (završava s razmišljanjem);
   u t=1 filozof 3 hoće jesti (ostali filozofi još dugo razmišljaju)

b) u t=0 svi filozofi hoće jesti, ali je filozof 2 prvi pozvao ČekajSem (drugi je filozof 4).

Završiti s prikazom nakon što prvi filozof koji jede završi i postavi oba semafora (i pritom možda propusti nekog drugog) ili ako se dogodi potpuni zastoj.

a) u t = 0 filozof 1 može zauzeti oba semafora:
   - za ČekajSEM(1) nema S* i S1 je slobodan, pa nema ograničenja
   - za ČekajSEM(2) S* je S1, ali ovaj filozof ga je zauzeo
   u t = 1 filozof 3 se blokira
   - ČekajSEM(3): S* je S1, filozof 3 nema veći prioritet!
   u t = 10 filozof 1 oslobađa semafore i onda tek može filozof 3
b) slično kao za a), samo što prvo kreće filozof 2, pa onda 4.

6. [3 boda] Mreža od pet računala spojena je u lanac R1$\longleftrightarrow$R2$\longleftrightarrow$R3$\longleftrightarrow$R4 (R1 komunicara samo s R2, R2 samo s R1 i R3, ...). Osmisliti jednostavan protokol koji se može koristiti za periodičku sinkronizaciju vremena (satova svih računala) u toj mreži, uz korištenje što manje razmjena podataka (i sa što manje podataka u razmjeni). Pretpostavka je da su satovi svih računala podjednako dobir/loši. Nije bitno s kojim se računalom satovi ostalih uspoređuju, ali je bitno da su oni jednaki. Pretpostaviti da prijenos podataka između dva računala jednako traje u oba smjera (ali trajanja među različitim računalima mogu biti različita). Računalo R1 uvijek inicira sinkronizaciju. Pokazati rad postupka ako su u početnom trenutku satovi s1=100, s2=102, s3=100, s4=103, prijenosi podataka traju: p12=5, p23=7, p34=3 te zadržavanja u svakom čvoru su po 1 jedinicu vremena.

```
Problem se može riješiti na nekoliko načina
(dovoljno je bilo riješiti na jedan način :))
S najmanjim brojem poruka ide: R1->R2->R3->R4->R3->R2->R1

1. svi se sinkroniziraju s R4, u povratnim porukama se prenose samo poruke
   s vremenima od R4 (i pamte trenuci slanja)
2. svi se sinkroniziraju s R4, u porukama se prenose sva vremena,
   ali pri računanju gledaju samo vremena slanja/primanja poruke
   s dotičnog čvora i čvora R4
3. svi se sinkroniziraju s idućim čvorom: R1 s R2, R2 s R3, R3 s R4,
   ali kaskadno: R1 šalje poruku R2, R2->R3, R3->R4,
   R4-> R3 - sada R3 može ažurirati svoj sat prema R4 (ima sva 4 vremena)
   onda R3 šalje trenutak primanja poruke od R2 i slanja prema R2, ali
   sada prema ažuriranom satu
   kad R2 primi poruku , ažurira svoj sat i šalje odgovor prema R1...

Pri slanju poruke natrag, obzirom na simetriju, dovoljno je poslati
i samo jednu vrijednost (t2+t3)/2 (ali su rješenju ipak slani oba t2 i t3)

Prikazano je rješenje za 1.
rt - vrijeme od početka slanja poruka, relativni "t"
[x] vrijeme u čvoru koji šalje ili prima poruku

        s1    <-5->    s2   <-7->   s3   <-3->   s4
rt=0:  [100]           102          100          103    R1 šalje prema R2
rt=5:   105           [107]         105          108    R2 prima od R1
rt=6    106           [108]         106          109    R2 šalje prema R3
rt=13   113            115         [113]         116    R3 prima od R2
rt=14   114            116         [114]         117    R3 šalje prema R4
rt=17   117            119          117         [120]   R4 prima od R3
rt=18   118            120          118         [121]   R4 šalje prema R3
u povratnoj poruci se šalje {t2=120 i t3=121}  (kraće bi bilo: t=120,5)
rt=21   121            123         [121]         124    R3 prima od R4
R3 sad ima: t1=114, t2=120, t3=121, t4=121 te računa
C=(t2+t3)/2-(t1+t4)/2=120,5-117,4=3 s3=s3+C=124  isti kao i R4
rt=22   122            124         [125]         125    R3 šalje prema R2
rt=29   129           [131]         132          132    R2 prima od R3
R2 sad ima: t1=108, t2=120, t3=121, t4=131 te računa
C=(t2+t3)/2-(t1+t4)/2=120,5-119,5=1 s2=s2+C=132  isti kao i R3 i R4
rt=30   130            133         [133]         133    R2 šalje prema R1
rt=35  [135]           138          138          138    R1 prima od R2
R1 sad ima: t1=100, t2=120, t3=121, t4=135 te računa
C=(t2+t3)/2-(t1+t4)/2=120,5-117,5=3 s1=s1+C=138 (s1=s2=s3=s4 !!!)
```

7. [1 bod] Neki arhitekt treba osmisliti neki SRSV i pritom treba odabrati operacijski sustav koji će se u njemu koristiti. Neka mu odabir može biti jedan on sljedećih: VxWorks, FreeRTOS, Linux s dodatkom CONFIG_PREEMPT_RT, običan Linux, obični Windowsi. U najkraćim crticama opišite prednosti i nedostatke svakog od navedenih operacijskih sustava za primjenu u SRSV-ima.

```
Uobičajeni kriteriji:
- podrška za SRSV sa strogim/blagim vremenskim ograničenjima
- promptna podrška programerima od strane proizvođača OS-a
- dostupnost programera (za tu platformu, koji se mogu brzo "priučiti")
- cijena sustava
```

OS	podržava SRSV-e	(SRSV) podrška programerima	dostupnost programera	cijena OS-a
VxWorks	stroge	kvalitetna	mala	licenca
FreeRTOS	stroge	mala (forumi)	mala	-
Linux+ patch	blage	mala (forumi)	velika	-
Linux	-	mala (forumi)	velika	-
Windowsi	-	mala (forumi)	velika	licenca

8. [1 bod] Osim ispravnosti u radu, gotovo svaki SRSV uključuje i sigurnosnu komponentu. Uz pretpostavku da se za neki sustav sigurnost ne može u potpunosti ostvariti zabranom pristupa, kako se ona može povećati korištenjem kriptografskih postupaka?

```
Korištenjem javnog/privatnog ključa može se potpisati svako dio koda
i onemogućiti izvođenje onog koji nije potpisan (koji nije "siguran").
U "kod" se ubraja i OS (i programi koji ga podižu - "bootloaderi").
Ključevi mogu biti pohranjeni u sklopovlju i do njih može biti "nemoguće"
doći (poznati su samo proizvođaču).
Također, moguće je zaštititi program od neovlaštenog kopiranja
(npr. dektiptira se samo prije pokretanja).
```

9. [1 bod] Razvoj programske potpore za SRSV se često obavlja u različitoj okolini (računalo, ...) od one za koju se priprema. U tom kontekstu usporediti korištenje virtualnih računala (i pripadajućih programa) s oblikom izolacije koji se nazivaju spremnici (npr. Linux Containeri). Koja je prednost jednih/drugih u fazama razvoja, ispitivanja, distribucije na ciljana računala, sigurnost, ...?

```
virtualna računala:
+ mogu simulirati nepostojeće sklopovlje
+ potpuna izolacija gostujućeg sustava
- veći zahtjevi na resurse (memoriju prvenstveno)
- manje performanse
- malo teža distribucija/deployment (veće slike, ...)

spremnici:
- programi moraju koristiti isti OS (jezgru)
- djelomična izolacija spremnika od OS-a i ostalih spremnika
+ manji zahtjevi na resurse
+ veće performanse
+ lakša distribucija/deployment
```