Blic 2A, OS, *student:*

1. Za koji od navedenih algoritama međusobnog isključivanja treba najmanje zajedničkih varijabli?

   a) Dekkerov b) Petersonov c) Lamportov

   d) algoritam ostvaren s instrukcijom *ispitaj i postavi* (TAS)

2. Početna dretva nekog programa stvara pet dretvi od kojih svaka izvodi kôd `dretva`. Početna dretva čeka da sve završe. Koju će vrijednost imati `a` na kraju?

```
int a=0;
void *dretva(void *x) {
  int i = 0;
  for(a=0; a<1000000; a++)
      i = i + 1;
  return NULL;
}
```

a) a = 0
b) a ≤ 1000000
c) a = 1000000
d) a = 5000000
e) a ≤ 5000000
f) 1000000 ≤ a ≤ 5000000

Blic 2A, OS: odgovori i komentari

1. TAS treba samo jednu varijablu ZASTAVICA => točan odgovor je **d)**
2. `for` ide po zajedničkoj varijabli `a`! varijabla `i` nije bitna – ne pita se što s njom

   Kada ne bi bilo problema s paralelnim radom `a++` tada bi sve dretve stale na 1000000

   Međutim, u zadnjoj iteraciji, nakon što su sve dretve napravile `a++`, može se dogoditi da pri ispitivanju uvjeta `a<1000000` više dretvi dohvati `a=999999` i prođe uvjet, uđu u petlju te na kraju povećaju `a++`. Ako to sad ne ide paralelno `a` se može povećati i više puta (999999 + 5 = 1000004). Tako da a može imati vrijednosti od 1000000 (minimalne) do 1000004 (maksimalne).

   Stoga je točan odgovor **f)** 1000000 ≤ a ≤ 5000000

   Priznat je i odgovor **c)** a = 1000000 jer je najbliže točnom rješenju i odbacuje vrijednosti manje od 1000000

Blic 2B, OS, *student:*

1. Koji algoritmi međusobnog isključivanja ne rade za više od dvije dretve?

   a) Dekkerov b) Petersonov c) Lamportov

   d) algoritam ostvaren s instrukcijom *ispitaj i postavi* (TAS)

2. Početna dretva nekog programa stvara pet dretvi od kojih svaka izvodi kôd `dretva`. Početna dretva čeka da sve završe. Koju će vrijednost imati `a` na kraju?

```
int a=0, i=0;
void *dretva(void *x) {
  for(i=0; i<1000000; i++)
      a = a + 1;
  return NULL;
}
```

a) a = 0
b) a ≤ 1000000
c) a = 1000000
d) a = 5000000
e) a ≤ 5000000
f) 1000000 ≤ a ≤ 5000000

Blic 2B, OS: odgovori i komentari

1. a) Dekkerov b) Petersonov
2. `for` ide po zajedničkoj varijabli `i`!

   Kada ne bi bilo problema s paralelnim radom `a=a+1` te `i++` tada bi sve dretve ukupno odradile najviše 1000000 iteracija. „Najviše“ zato jer u `for` imamo `i=0`, pa dretve koje kasnije kreću će poništiti prethodna povećanja. Tada bi vrijedilo b)

   Međutim, dretve paralelno rade. Operacija `a=a+1` može uzrokovati da konačna vrijednost varijable `a` bude i manja.

   S druge strane, paralelni rad `i++` može uzrokovati da se broj iteracija petlje poveća (jer se i ponekad ne povećava u svakoj dretvi). Koliko se puta petlja teoretski može povećati u najgorem slučaju? Jako puno puta! Evo primjera scenarija sa samo dvije dretve: obje dretve krenu raditi u petlji, ali prva stane (OS ju privremeno prekine) na a=a+1, tj. i=0 za nju. Druga odradi sve do zadnje iteracije, ali ne i zadnju. Prva tada napravi jednu iteraciju i poveća i na jedan. Druga sada opet kreće, ali i za nju je sada i=1 pa opet radi puno iteracija. Itd. Uz više dretvi to je i složenije. Odgovor nije beskonačno, ali nije tako lako doći do prave vrijednosti.

   Od ponuđenih odgovora najpreciznija i najrealnija je **e)** iako su priznati i **b)**, **c)** i **f)**

Blic 2C, OS, *student:*

1. Koji algoritmi međusobnog isključivanja imaju problem radnog čekanja?

   a) Dekkerov b) Petersonov c) Lamportov

   d) algoritam ostvaren s instrukcijom *ispitaj i postavi* (TAS)

2. Početna dretva nekog programa stvara pet dretvi od kojih svaka izvodi kôd `dretva`. Početna dretva čeka da sve ostale završe te potom ispiše vrijednost globalne varijable `a`. Koju će vrijednost imati `a` na kraju?

```
int a=0;
void *dretva(void *x) {
  int a=0, i;
  for(i=0; i<1000000; i++)
      a = a + 1;
  return NULL;
}
```

a) a = 0
b) a ≤ 1000000
c) a = 1000000
d) a = 5000000
e) a ≤ 5000000
f) 1000000 ≤ a ≤ 5000000

Blic 2C, OS: odgovori i komentari

1. svi
2. U funkciji dretva varijabla `a` je lokalna, tako da se globalna varijabla a neće mijenjati i imati će vrijednost nula (odgovor **a**)

   Oni koji dovoljno ne poznaju C su mogli to previdjeti, pa je 0,5 boda dano i za odgovor **e)** koji bi bio točan kada ispred `a=0` u funkciji ne bi stajalo `int`.

Blic 2D, OS, *student:*

1. Koji algoritam međusobnog isključivanja od navedenih se ostvaruje s najmanjim brojem instrukcija?

   a) Dekkerov b) Petersonov c) Lamportov

   d) algoritam ostvaren s instrukcijom *ispitaj i postavi* (TAS)

2. Početna dretva nekog programa stvara pet dretvi od kojih svaka izvodi kôd `dretva`. Početna dretva čeka da sve ostale završe te potom ispiše vrijednost globalne varijable `a`. Koju će vrijednost imati `a` na kraju?

```
int a=0, i;
void *dretva(void *x) {
  int a=0;
  for(i=0; i<1000000; i++)
      a = a + 1;
  return NULL;
}
```

a) a = 0
b) a ≤ 1000000
c) a = 1000000
d) a = 5000000
e) a ≤ 5000000
f) 1000000 ≤ a ≤ 5000000

Blic 2D, OS: odgovori i komentari

1. TAS treba samo dvije/tri instrukcije => točan odgovor je **d)**
2. U funkciji dretva varijabla `a` je lokalna, tako da se globalna varijabla a neće mijenjati i imati će vrijednost nula (odgovor **a**)

   Oni koji dovoljno ne poznaju C su mogli to previdjeti, pa je 0,5 boda dano i za odgovor **e)** koji je „najtočniji“ kada ispred `a=0` u funkciji ne bi stajalo `int`. tj. kada bi zadak bio kao skoro kao i 2B.