Programiranje 3
izpit

23. junij 2016

You must solve the exam on your own. The time of writing is 64 minutes 25 seconds.

Good luck!

IME IN PRIIMEK: ______________________

ŠTUDENTSKA ŠTEVILKA: ______________________

DATUM: ______________________

PODPIS: ______________________

1. naloga: Specify the definition of one-time property, and for each command in the program below, check if this property stands:

```
int x = 0, y = 10, z = 6
co x = x + z;  (1)       x = x+z (1') NO
|| y = x - y;  (2)       y = x-y  (2') YES
|| y = 9;      (3)       z = 9     (3') YES
oc
```

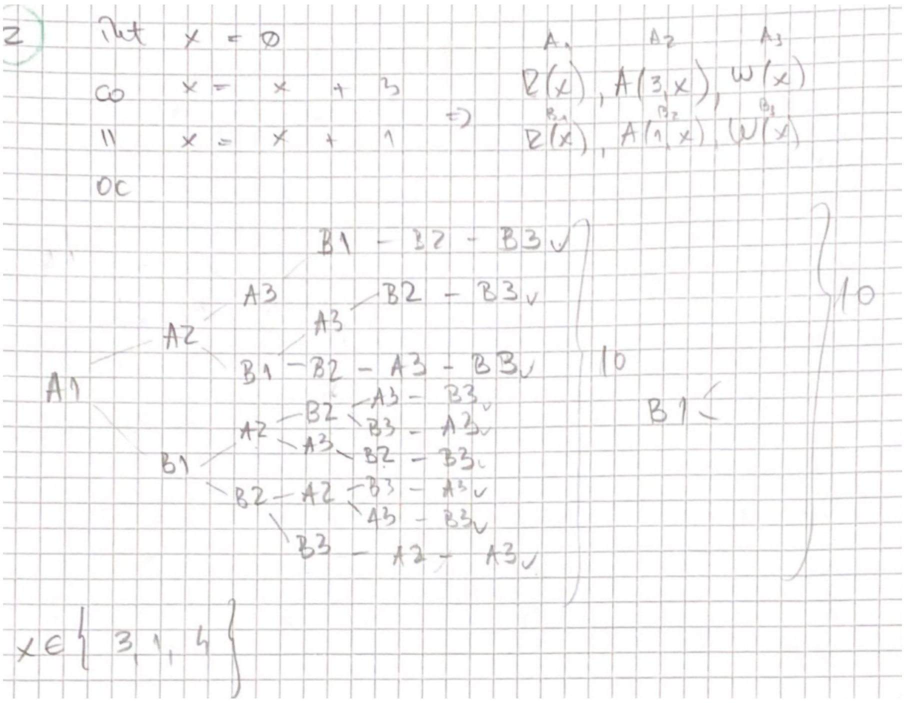
At most once property:
if x := e;
(1) e has at one critical reference and x is not read/written by another process.
(2) e does not include critical references, x can be read/written by many processes.

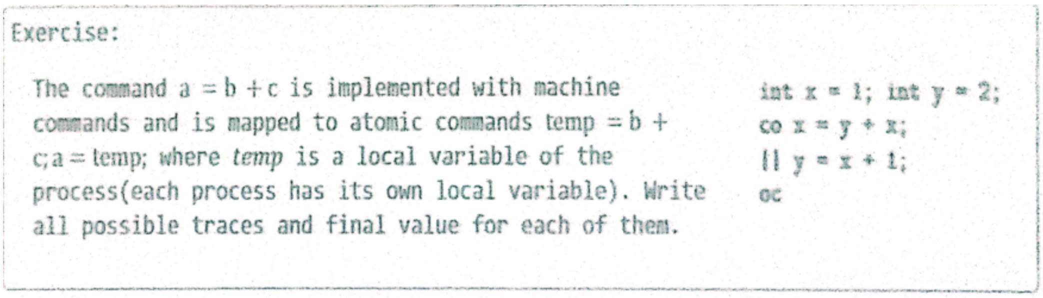
(1) has no critical reference => property stands (x isn't touched by other)
(2) has 2 critical references => doesn't stand
(3) property stands. (z isn't touched by other.)
----
(1') no, since critical ref to z and x read by (2')
(2') yes holds, since has an CR and not read by other
----
(1") x = x+z   No, read by 2", refs z.
(2") y = x-y   No, read by 3", refs x.
(3") z = 1+y   No, read by 1", refs y.

2. naloga:

```
int x = 0;
co x = x + 3;
|| x = x + 1;
oc
```

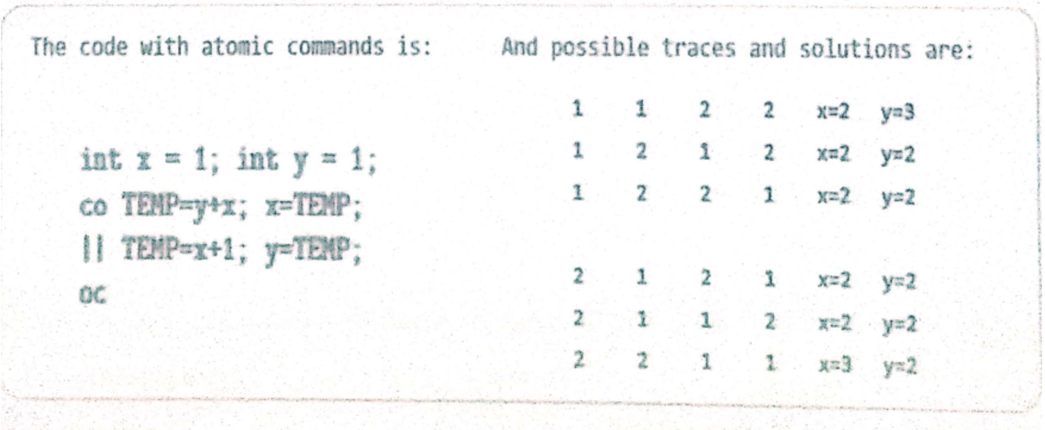
Command A = B + C is implemented with machine instructions; it is mapped to the atomic instruction TEMP = B + C; A = TEMP; where TEMP is the local process variable (so each process has its own). List all possible program traces and give the final value for each trace.




Exercise:

The command a = b + c is implemented with machine commands and is mapped to atomic commands temp = b + c; a = temp; where *temp* is a local variable of the process(each process has its own local variable). Write all possible traces and final value for each of them.

```
int x = 1; int y = 2;
co x = y + x;
|| y = x + 1;
oc
```


The code with atomic commands is:

```
int x = 1; int y = 1;
co TEMP=y+x; x=TEMP;
|| TEMP=x+1; y=TEMP;
oc
```

And possible traces and solutions are:

1	1	2	2	x=2	y=3
1	2	1	2	x=2	y=2
1	2	2	1	x=2	y=2
2	1	2	1	x=2	y=2
2	1	1	2	x=2	y=2
2	2	1	1	x=3	y=2

3. naloga: Use the semaphores to implement a p process barrier that is suitable for multiple use. You can not change the global initialisation of the program - you can only program the parallel process code (all processes implement the same code). The initialization of global variables is given as:

```
int x = p;
semafor s1 = 1;
semafor s2 = 0;
```

```
int x = p;
semafor s1 = 1;
semafor s2 = 0;

P(S1);
x--;
if(x==0){
   for(x=0; x<p; x++){
      V(S2);
   }
}
```

4. naloga:

```
int x = p;
int x = 0; sem s1 = 1, s2 = 0;
co P(s2); P(s1); x = x * 7; V(s1);      (1)
// P(s1); x = x * 6 + x; V(s2); V(s1);(2)
// P(s1); x = x + 4; V(s1);             (3)
oc
```

What are the possible final values for x? Comment on each step!

(1) stops since s2=0.
We can start with either (2) or (3).

if we start from (2), then we should continue with (1) - (3) which is x = 4;
OR we can start from (2) and continue with (3)-(1) so x=28;
OR we can start from (3) and continue with (2)-(1) so x=196;