

Sva teorijska pitanja ikad postavljena na  
završnim ispitima iz PROGI (OPP) u zadnjih par  
godina

sretno!!!!!!

OPP\_ZI\_2018\_2019\_GrupaB  
\_R

Generičke aktivnosti procesa programskog inženjerstva mogu se podijeliti u četiri osnovne skupine. Navedite ih prema vremenskom redoslijedu.

Generičke aktivnosti procesa programskog inženjerstva mogu se podijeliti u četiri osnovne skupine. Navedite ih prema vremenskom redoslijedu.

Rj. Specifikacija, oblikovanje i implementacija, validacija i verifikacija, evolucija.

Navedite tipične načine opisa  
zahtjeva sustava.

# Navedite tipične načine opisa zahtjeva sustava.

Rj. strukturiranim prirodnim jezikom,  
specijalnim jezikom za opis oblikovanja  
(npr. SDL), grafičkom notacijom (npr.  
UML) i matematičkom specifikacijom  
(FSM, teorija skupova, logika).

Što je to projektni dnevnik zaostataka (engl. product backlog) kod pristupa SCRUM radnog okvira? Tko je odgovoran za sadržaj, raspoloživost i sortiranje dnevnika zaostataka?

Što je to projektni dnevnik zaostataka (engl. product backlog) kod pristupa SCRUM radnog okvira? Tko je odgovoran za sadržaj, raspoloživost i sortiranje dnevnika zaostataka?

Rj. strukturiranim prirodnim jezikom, specijalnim jezikom za opis oblikovanja (npr. SDL), grafičkom notacijom (npr. UML) i matematičkom specifikacijom (FSM, teorija skupova, logika).



Kako se naziva princip dobrog oblikovanja programske potpore kod kojeg se traži grupiranje međusobno povezanih elemenata, a sve ostale elemente se stavlja izvan grupe?

Kako se naziva princip dobrog oblikovanja programske potpore kod kojeg se traži grupiranje međusobno povezanih elemenata, a sve ostale elemente se stavlja izvan grupe?

Rj. Povećanje kohezije

Koja su 3 osnovna tipa vidljivosti u UML-dijagramu razreda, kako ih označavamo i što ona znače?

# Koja su 3 osnovna tipa vidljivosti u UML-dijagramu razreda, kako ih označavamo i što ona znače?

Javni (public, +) – svi razredi mogu pristupiti atributu / metodi

Zaštićeni (protected, #) – svi razredi iz hijerarhije podrazreda ili iz samog razreda mogu pristupiti tom atributu / metodi

Privatni (private, -) – samo je moguće pristupiti iz samog razreda tom atributu / metodi

Što je metoda u objektno orijentiranim programskim jezicima i kako se prikazuje poziv metode na UML-sekvencijskom dijagramu?

Što je metoda u objektno orijentiranim programskim jezicima i kako se prikazuje poziv metode na UML-sekvencijskom dijagramu?

Rj. Metoda je način izvođenja ili implementacija neke operacije. To je procedura, funkcija, rutina, proceduralna apstrakcija koja se koristi za implementaciju ponašanja razreda. Poziv metode prikazuje se strelicom s nazivom poruke (metoda je poziv ili slanje poruka).

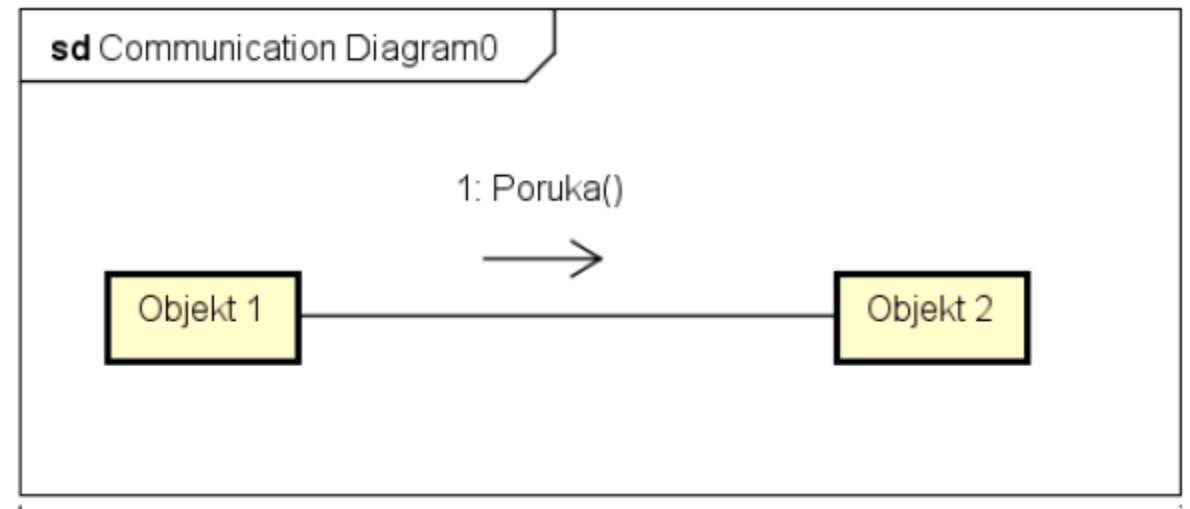
Definirajte alociranje  
odgovornosti (engl.  
responsibility) razredima.

# Definirajte alociranje odgovornosti (engl. responsibility) razredima.

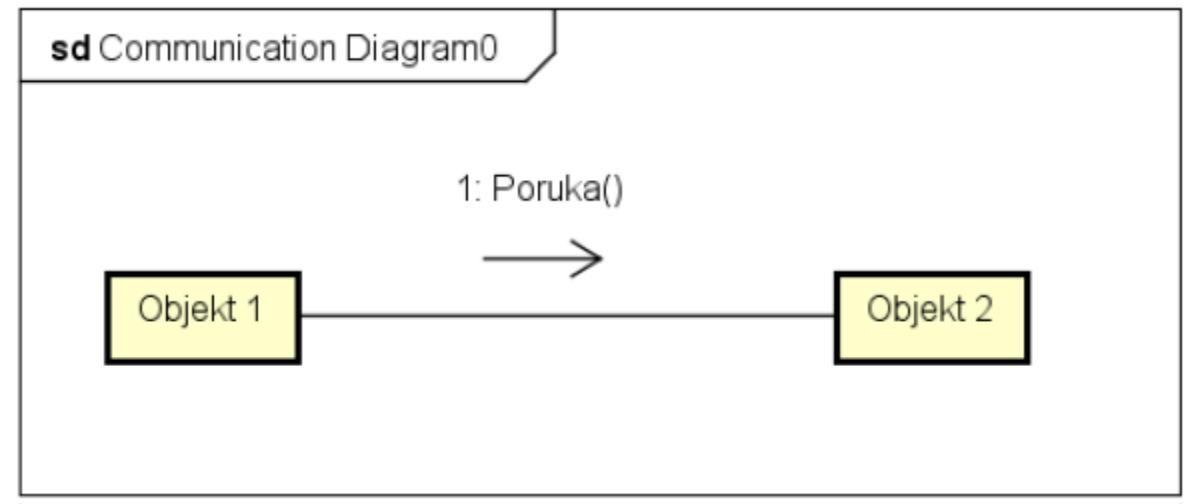
Rj. Odgovornost (engl. responsibility) je nešto što sustav mora izvršiti. Sve odgovornosti jednog razreda moraju biti jasno povezane. Ako jedan razred ima previše odgovornosti, razmotri podjelu toga razreda u različite razrede. Ako razred nema odgovornosti, tada je vjerojatno beskoristan. Ako se neka odgovornost ne može pripisati niti jednom od postojećih razreda, mora se kreirati novi razred.



Koji tip poruke je prikazan na komunikacijskom dijagramu na slici?



Koji tip poruke je prikazan na komunikacijskom dijagramu na slici?



Rj. Asinkrona poruka  
(engl. asynchronous message)

Objasnite što su značke i kako se kreću u UML dijagramu aktivnosti (engl. activity diagram).

# Objasnite što su značke i kako se kreću u UML dijagramu aktivnosti (engl. activity diagram).

Rj. Značke su dio semantike bez grafičkog prikaza. Značka može predstavljati: upravljački tijek; objekt; podatak. Kreću se od izvorišta prema odredištu vezama ovisno o: ispunjenim uvjetima izvornog čvora, postavljenim uvjetima veza (engl. Edge guard conditions), preduvjetima ciljnog čvora.

Kod kojeg je arhitekturnog obrasca  
(engl. architectural pattern) jedna  
od temeljnih značajki implicitno  
pozivanje komponenata?

Kod kojeg je arhitekturnog obrasca (engl. architectural pattern) jedna od temeljnih značajki implicitno pozivanje komponenata?

Rj: Kod arhitekture zasnovane na događajima (engl. event-based architecture).

Koji je cilj provođenja aktivnosti  
ispitivanja u procesu oblikovanja  
programske potpore?

# Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

Rj. Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema ispitivane programske podrške.

Priznaje se i kraće rješenje: pronalaženje pogrešaka.



Za funkcijsko ispitivanje ekvivalentnim particijama (engl. equivalence partition):

- a) Navedite korake ispitivanja.
- b) Na primjeru jednog ulaznog 3-znamenkastog broja u intervalu  $[700, 997]$ , navedite potrebne minimalne ispitne slučajeve.

Za funkcijsko ispitivanje ekvivalentnim particijama (engl. equivalence partition):

Rj.

a) Navedite korake ispitivanja.

b) Na primjeru jednog ulaznog 3-znamenkastog broja u intervalu [700, 997], navedite potrebne minimalne ispitne slučajeve.

1. **Odredi particije za sve ulazne varijable.**
2. **Za sve particije odaberi vrijednosti ispitivanja.**
3. **Definiraj ispitne slučajeve koristeći odabrane vrijednosti.**
4. **Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.**

Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	699	ne prolazi
IS2	700	prolazi
IS3	801	prolazi
IS4	997	prolazi
IS5	998	ne prolazi

\*priznaju se i dodatni ispitni slučajevi: (npr. 500, ne prolazi) (npr. 1200, ne prolazi)

14. (2 boda) Za sljedeću funkciju:

- a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.
- b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

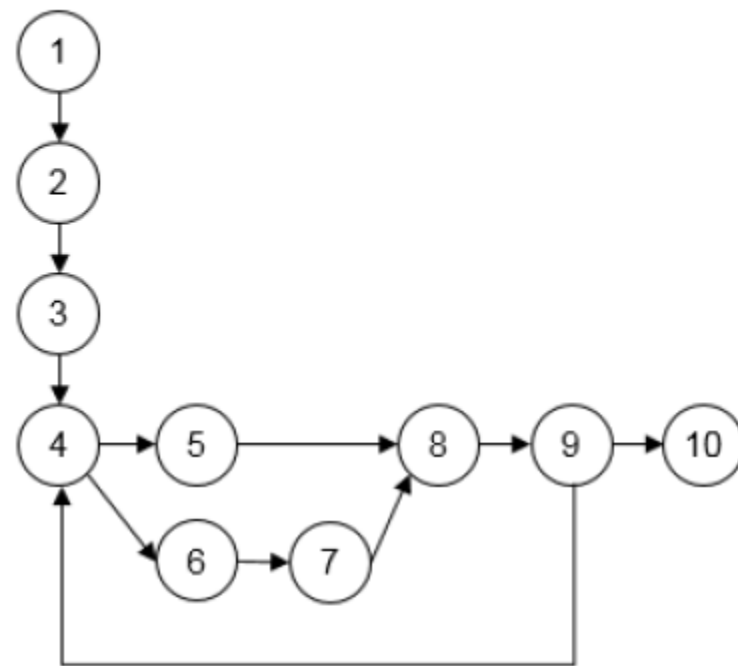
```
public void TDM(A[], x, y , z , B[])
{
    int i = x;
    int j = y;
    int k = x;
    do {
        if (i < y && (j >= z && A[i] <= A[j])) {
            B[k] = A[i++];
        } else {
            B[k] = 4+A[j]/2;
            j = j++;
        }
        k++;
    } while (k < z);
    return;
}
```

Rj.

a) Ubaciti novo rješenje!!

```
public void TDM(A[], x , y , z , B[])
{
    int i = x ; (1)
    int j = y ; (2)
    int k = x;   (3)

    do {
        if (i < y && (j >= z || A[i]<= A[j])) (4) {
            B[k] = A[i++]; (5)
        } else {
            B[k] = 4+A[j]/2; (6)
            j = j++; (7)
        }
        k++; (8)
    } while (k < z); (9)
    return; (10)
}
```



b)  $CV(G) = 11 - 10 + 2 \cdot 1 = 3$

Definirajte logičku posljedicu  
 $(\Gamma \models \phi)$  putem modela ili putem  
teorema o dedukciji.

Definirajte logičku posljedicu  
 $(\Gamma \models \phi)$  putem modela ili putem  
teorema o dedukciji.

Rj.  $\phi$  je logička posljedica skupa formula  $\Gamma$   
ako je svaki model od  $\Gamma$  ujedno i model od  
 $\phi$  ili  $\phi$  je logička posljedica skupa formula  
 $\Gamma$  ako i samo ako je  $(\Gamma \Rightarrow \phi)$  tautologija  
(priznaje se i:  $\Gamma \wedge \neg \phi$  kontradikcija)

Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:

"U poslužitelju Sparc T8-4 postoje barem dva procesora."

Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:  
"U poslužitelju Sparc T8-4 postoje barem dva procesora."

Rj. poslužitelj(x) - x je poslužitelj

procesor(x) - x je procesor

postoji\_u(x,y) – x postoji u y

=(x,y) - x je jednako y

Sparc T8-4 - konstanta

$\text{poslužitelj}(\text{Sparc T8-4}) \Rightarrow \exists x \exists y (\text{procesor}(x) \wedge \text{procesor}(y) \wedge \text{postoji\_u}(x, \text{Sparc T8-4}) \wedge \text{postoji\_u}(y, \text{Sparc T8-4}) \wedge \neg \text{=(x,y)})$



Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic):

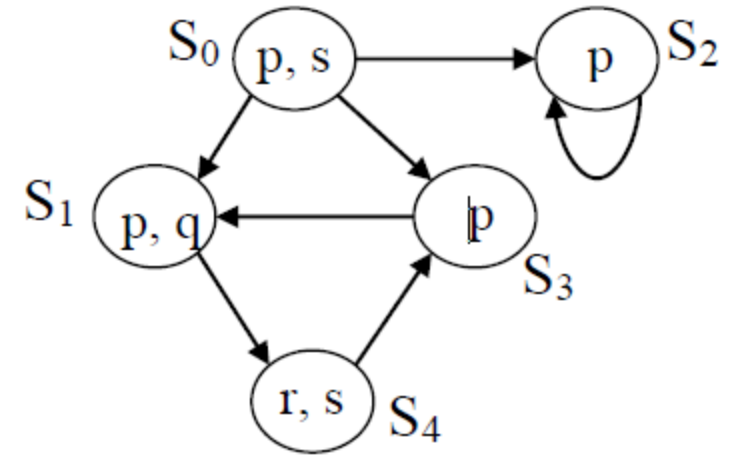
"U svakom stanju nakon početnog stanja (ali ne i u početnom stanju) postoji put na kojem cijelo vrijeme vrijedi  $\text{stop}=1$ ."

Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic):

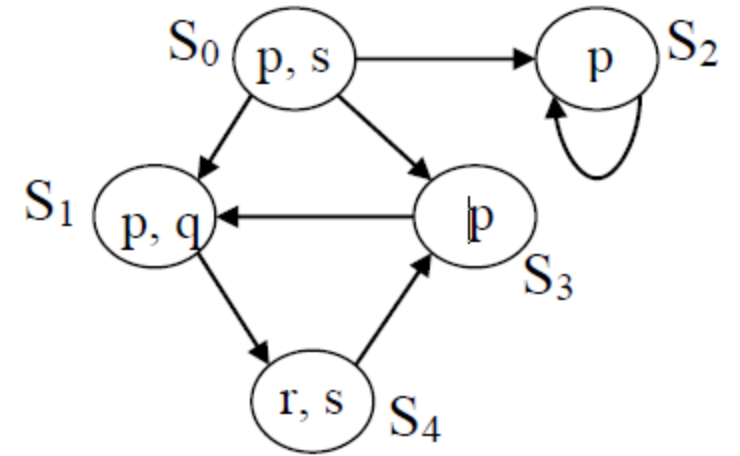
"U svakom stanju nakon početnog stanja (ali ne i u početnom stanju) postoji put na kojem cijelo vrijeme vrijedi  $\text{stop}=1$ ."

Rj.  $AX\ EG\ (\text{stop}=1)$ , priznaje se i:  
 $AG\ AX\ EG\ (\text{stop}=1)$ , jer se moglo i  
tako shvatiti

Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu **EG (p)**.



Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu **EG (p)**.



Takva stanja su =  $S_0, S_2$

Za arhitekturu protoka podataka,  
navedite i objasnite podjelu prema  
vrsti izvršavanja obrade podataka.

# Za arhitekturu protoka podataka, navedite i objasnite podjelu prema vrsti izvršavanja obrade podataka.

## **Rj. Skupno – sekvencijska (engl. Batch-sequential)**

Podaci se između koraka (podsustava za obradu) prenose u cijelosti

Svaki podsustav se izvodi do kraja prije prelaska na idući korak

Svaki podsustav je neovisan o drugima

## **Cjevovodi i filteri (engl. Pipes-and-filters)**

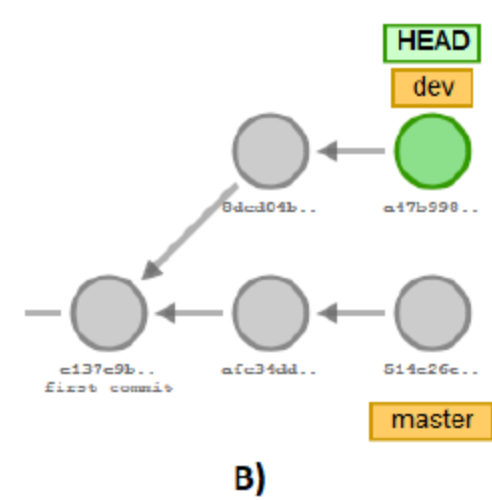
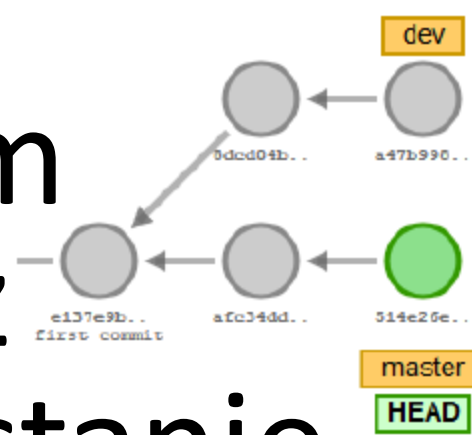
Inkrementalna, konkurentna transformacija podataka kroz korake (filtere), čim podaci postanu dostupni

Komponente: izvori podataka, filteri, cjevovodi i ponori podataka

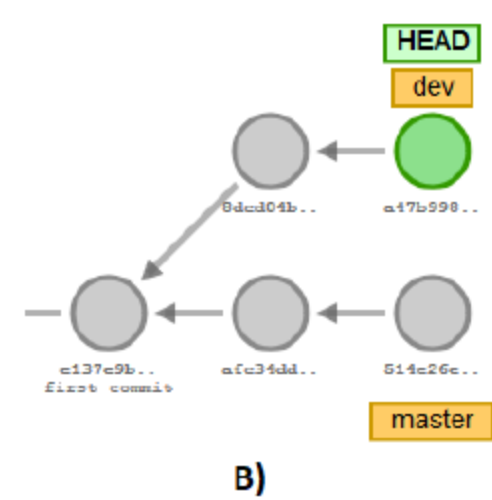
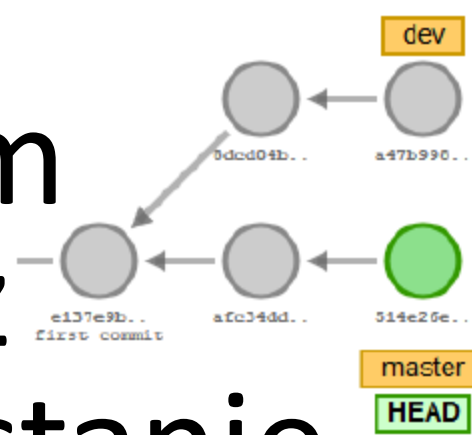
Filteri su neovisni o ostalim filterima, ne pamte stanje

Cjevovodi su FIFO međuspremnici u obliku U/I tokova podataka (input/output stream)

Kojom git naredbom  
repozitorij prelazi iz  
stanja sa slike A) u stanje  
sa slike B)?



Kojom git naredbom  
repozitorij prelazi iz  
stanja sa slike A) u stanje  
sa slike B)?



git checkout dev



OPP\_ZI\_2016\_rijesen

Navedite generičke aktivnosti u svim procesima programskog inženjerstva.

Navedite generičke aktivnosti u svim procesima programskog inženjerstva.

Rješenje: Specifikacija, oblikovanje i implementacija, validacija i verifikacija, evolucija

Kako se nazivaju programski alati koji podupiru aktivnosti procesa programskog inženjerstva?

Kako se nazivaju programski alati koji podupiru aktivnosti procesa programskog inženjerstva?

Rješenje: CASE , alati Računalom podržano programsko inženjerstvo ...

Navedite barem dvije značajke metodologije ubrzanog razvoja (engl. agile methodology) programske potpore.

Navedite barem dvije značajke metodologije ubrzanog razvoja (engl. agile methodology) programske potpore.

Rješenje: Iterativni razvoj, mali inkrementi, kontinuirano poboljšanje PP, naglasak na ljude i suradnju, uključenost korisnika u proces razvoja...

Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili korisničke zahtjeve.



Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili korisničke zahtjeve.

Rješenje: UC-ovi s pripadajućim dijagramima (funkcionalni zahtjevi), sekv. dijagrami, lista nefunkcionalnih zahtjeva

Što su zahtjevi domene  
primjene?

# Što su zahtjevi domene primjene?

Rješenje: Funkc. i nefunkc. zahtjevi koji proizlaze iz domene primjene/specifični su za domenu primjene/karakteriziraju domenu primjene

Koji se modifikatori vidljivosti operacija koriste u OO paradigmi kako bi se postigla enkapsulacija? Prikažite kako se označavaju na UML-dijagramima razreda.

Koji se modifikatori vidljivosti operacija koriste u OO paradigmi kako bi se postigla enkapsulacija? Prikažite kako se označavaju na UML-dijagramima razreda.

Rješenje: Public +, private -, protected #... mogu navesti i package (ali ne moraju)

Koji minimalni uvjet mora ispuniti neki razred da bi bio apstraktni razred?

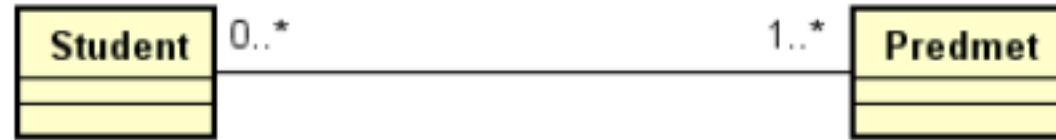
Može li se instancirati takav razred?

Koji minimalni uvjet mora ispuniti neki razred da bi bio apstraktni razred?

Može li se instancirati takav razred?

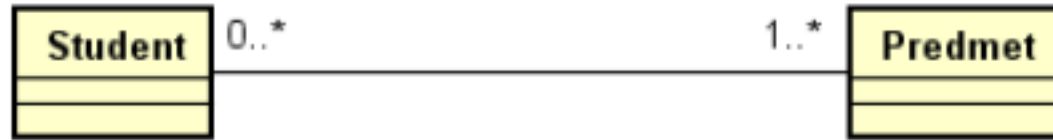
Rješenje: Razred koji ima barem jednu apstraktnu metodu. Ne može.

Prema donjem dijagramu razreda odgovorite  
koliko najviše predmeta  
može svaki student upisati te mora li uopće  
upisati predmet?





Prema donjem dijagramu razreda odgovorite koliko najviše predmeta može svaki student upisati te mora li uopće upisati predmet?



Rješenje: Student može upisati neograničeni (nedefinirani) broj predmeta, ali mora upisati barem 1.

Nacrtajte dijagram komponenti s dvije komponente koje su povezane jednim sučeljem. Navedite i objasnite vrste sučelja komponenti.

# Nacrtajte dijagram komponenti s dvije komponente koje su povezane jednim sučeljem. Navedite i objasnite vrste sučelja komponenti.

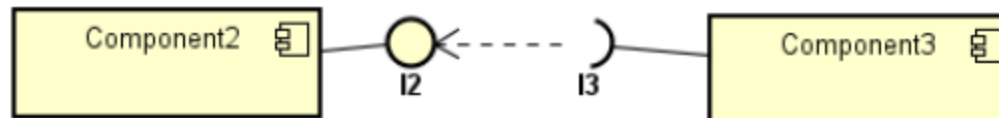
*Rješenje:*

*Ponudeno sučelje (eksportirano sučelje) – skup usluga/metoda koje komponenta realizira (nudi).*

*Zahtijevano sučelje (importirano sučelje) – skup usluga/metoda koje komponenta koristi/poziva od druge komponente.*



*Može i ovako*



Navedite i kratko opišite  
elemente UML-dijagrama  
razmještaja.

Navedite i kratko opišite  
elemente UML-dijagrama  
razmještaja.

Rješenje: Čvorovi - uređaji i okolina  
izvođenja; Komponente (programski  
artefakti); Spojevi (veze) između  
čvorova i između komponenti

Kako je organiziran primjenski program u uslužnoj arhitekturi?

# Kako je organiziran primjenski program u uslužnoj arhitekturi?

Rješenje: Uslužno usmjerena arhitektura organizira primjenski program (cjelovitu aplikaciju) kao kolekciju usluga koje međusobno komuniciraju uporabom dobro definiranih sučelja

Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)?



# Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)?

*Rješenje:*

<i>Naziv</i>	<i>Razumljivo ime</i>
<i>Opis</i>	<i>Opis problema</i>
<i>Rješenje</i>	<i>Opis predložka koji može biti upotrijebljen na različite načine</i>
<i>Posljedice</i>	<i>Rezultati i pogodnosti primjene obrasca uporabe</i>

Navedite kojoj skupini ili kojim skupinama pripadaju metode u razredu AbstractClient OCSF arhitekture koje se ne redefiniraju.

#### ***AbstractClient***

```
<<control>>  
openConnection ()  
sendToServer ()  
closeConnection ()  
<<hook>>  
connectionEstablished ()  
connectionClosed ()  
connectionException ()  
<<slot>>  
handleMessageFromServer ()  
<<accessor>>  
isConnected()  
getPort ()  
setPort ()  
getHost ()  
setHost ()  
getInetAddress ()
```

Navedite kojoj skupini ili kojim skupinama pripadaju metode u razredu AbstractClient OCSF arhitekture koje se ne redefiniraju.

Rješenje: <<control>> i  
<<accessor>>

#### **AbstractClient**

```
<<control>>  
openConnection ()  
sendToServer ()  
closeConnection ()  
<<hook>>  
connectionEstablished ()  
connectionClosed ()  
connectionException ()  
<<slot>>  
handleMessageFromServer ()  
<<accessor>>  
isConnected()  
getPort ()  
setPort ()  
getHost ()  
setHost ()  
getInetAddress ()
```

Za slučaj arhitekture klijent-poslužitelj kada postoji n spojenih klijenata

izračunajte minimalan broj dretvi pri radu poslužitelja implementiranog

objektno usmjerenim radnim okvirom OCSF (NAPOMENA: zanemarite

administracijske dretve, dretve OS-a, VM,...).

Za slučaj arhitekture klijent-poslužitelj kada postoji  $n$  spojenih klijenata

izračunajte minimalan broj dretvi pri radu poslužitelja implementiranog

objektno usmjerenim radnim okvirom OCSF (NAPOMENA: zanemarite

administracijske dretve, dretve OS-a, VM,...).

**Rješenje:  $n+1$**

Koji je cilj provođenja aktivnosti  
ispitivanja u procesu oblikovanja  
programske potpore?

# Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

Rješenje: Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja

pronalaženjem kvarova i problema ispitivane programske podrške.

Priznaje se i kraće rješenje: pronalaženje pogrešaka.

Na koji način možemo odrediti broj mogućih putova koji minimalno jednom pokrivaju izvođenje svih naredbi i uvjeta, uz pretpostavku dostupnosti izvornog koda?



Na koji način možemo odrediti broj mogućih putova koji minimalno jednom pokrivaju izvođenje svih naredbi i uvjeta, uz pretpostavku dostupnosti izvornog koda?

Rješenje: Korištenjem teorije grafova – izračun broja linearno neovisnih putova – ciklomatska složenost  $CV(G) = \text{Lukovi} - \text{Čvorovi} + 2 * P$

Primjenom tehnike kombinacijskog ispitivanja (engl. Combination testing) potrebno je ispitati funkciju koja kao ulaze prima tri dvoznamenkasta cijela broja koji se unose tipkovnicom. Izračunajte broj potrebnih ispitnih slučajeva.

Primjenom tehnike kombinacijskog ispitivanja (engl. Combination testing) potrebno je ispitati funkciju koja kao ulaze prima tri dvoznamenkasta cijela broja koji se unose tipkovnicom. Izračunajte broj potrebnih ispitnih slučajeva.

Rješenje: ulazne vrijednosti -99..99

$$199 * 199 * 199 = 7\,880\,599$$

bilo bi izvrsno da studenti prokomentiraju i neispravne vrijednosti ulaza.

Navedite elemente  
dokumentacije ispitnih slučajeva  
(engl. test case).

# Navedite elemente dokumentacije ispitnih slučajeva (engl. test case).

Rješenje: Minimalno: Ulaz, Očekivani izlaz, Stvarni rezultat

Dodatno: ime (naziv) ispitnog slučaja, status (anomalije), kako ispitivati modul ili funkciju, opis stanja prije ispitivanja, funkcije koja se ispituje...

U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GIT sustav za upravljanje inačicama

datoteka (engl. Version control, revision control, source control).

Gdje su

smješteni podaci koje mijenjaju pojedini članovi tima? Obrazložite.

U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GIT sustav za upravljanje inačicama

datoteka (engl. Version control, revision control, source control).

Gdje su

smješteni podaci koje mijenjaju pojedini članovi tima? Obrazložite.

Rješenje: u lokalnom repozitoriju.  
Članovi tima mijenjaju podatke u svojim  
lokalnim repozitorijima.

Formalan logički sustav je uređeni par  $(\Gamma, L)$ . Kako definiramo model formalnog logičkog sustava?



Formalan logički sustav je uređeni par  $(\Gamma, L)$ . Kako definiramo model formalnog logičkog sustava?

Rješenje: Neka interpretacija je model FLS ako evaluira sve njegove formule  $\Gamma$  u istinito.

Definirajte potrebne predikate i konstante te preslikajte rečenice u dobro definirane formule predikatne logike prvoga reda:

"Neki ispravan i kompletan formalan sustav može biti neodlučljiv."

Propozicijska logika je formalan sustav koji je ispravan, kompletan i odlučljiv."

Definirajte potrebne predikate i konstante te preslikajte rečenice u dobro definirane formule predikatne logike prvoga reda:

"Neki ispravan i kompletan formalan sustav može biti neodlučljiv."

Propozicijska logika je formalan sustav koji je ispravan, kompletan i odlučljiv."

*Rješenje:*

*formalan\_sustav(x) - x je formalan sustav*

*ispravan(x) - x je ispravan*

*kompletan(x) - x je kompletan*

*odlučljiv(x) - x je odlučljiv*

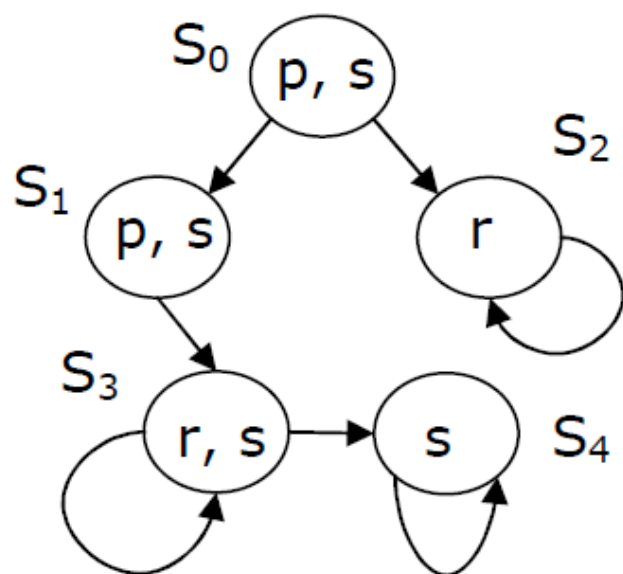
*Propozicijska\_logika = konst.*

*$\exists x (formalan\_sustav(x) \wedge ispravan(x) \wedge kompletan(x) \wedge !odlučljiv(x))$*

*$formalan\_sustav(Propozicijska\_logika) \wedge ispravan(Propozicijska\_logika) \wedge$   
 $kompletan(Propozicijska\_logika) \wedge odlučljiv(Propozicijska\_logika)$*

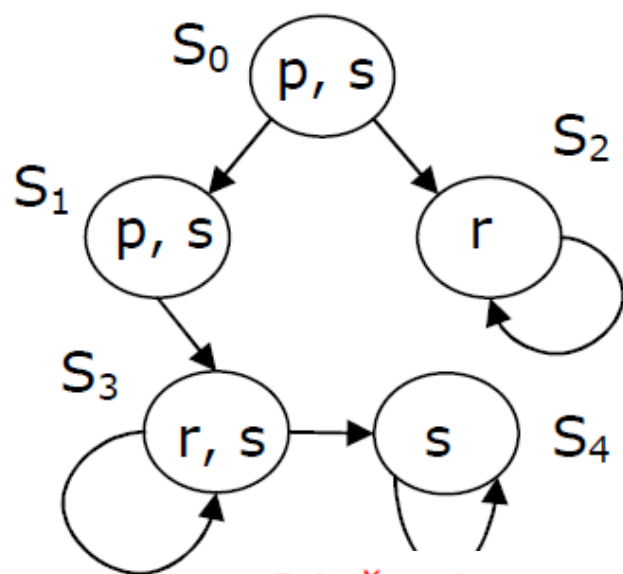
**22. (2 boda)** Za zadani model implementacije  $M$  Kripke strukturom prema slici potrebno je:

- Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).
- Odrediti sva stanja koja zadovoljavaju formulu  $\mathbf{A} (p \mathbf{U} (r \wedge s)).$



**22. (2 boda)** Za zadani model implementacije M Kripke strukturom prema slici potrebno je:

- Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).
- Odrediti sva stanja koja zadovoljavaju formulu **A** ( $p \mathbf{U} (r \wedge s)$ ).



*Rješenje:*

*a)*

$$S = \{S_0, S_1, S_2, S_3, S_4\}$$

$$R = \{(S_0, S_1), (S_0, S_2), (S_2, S_2), (S_1, S_3), (S_3, S_3), (S_3, S_4), (S_4, S_4)\}.$$

$$L(S_0) = \{p, s\}; L(S_1) = \{p, s\}; L(S_2) = \{r\}; L(S_3) = \{r, s\}; L(S_4) = \{s\}$$

*b)  $S_1$  i  $S_3$*

OPP\_ZI\_2017\_2018\_grupaB

Navedite najmanje četrir vrste  
projekata razvoja programske  
potpore.

Navedite najmanje četiri vrste projekata razvoja programske potpore.

Korektivni, adaptivni, unapređujući, reinženjerstvo, potpuno novi projekti, integrativni, hibridni



Navedite metode izlučivanja  
korisničkih zahtjeva.

Navedite metode izlučivanja  
korisničkih zahtjeva.

Rj. intervjuiranje, scenarij, obrasci  
uporabe, dinamičke interakcije  
korištenjem sekvencijskih  
dijagrama

Koje su tri vrste sudionika u  
Scrum timu?

# Koje su tri vrste sudionika u Scrum timu?

Rj. vlasnik proizvoda (product owner), razvojni tim (development team), scrum vođa (scrum master)

Rj. vlasnik proizvoda (product owner), razvojni tim (development team), scrum vođa (scrum master)

Rj. vlasnik proizvoda (product owner), razvojni tim (development team), scrum vođa (scrum master)

Rj. Konceptijska, logička, izvršna.

Objasnite princip dobrog oblikovanja programske podpore: oblikuj konzervativno (engl. design defensively).

Objasnite princip dobrog oblikovanja programske potpore: oblikuj konzervativno (engl. design defensively).

Rj. Ne koristiti pretpostavke kako će netko upotrebljavati oblikovanu komponentu obraditi sve slučajeve u kojima se komponenta može neprikladno upotrijebiti provjeriti valjanost ulaza u komponentu provjerom definiranih pretpostavki.



6. (2 boda) Razred BankovnaUsluga i sučelje IRed opisani su sljedećim kôdom:

```
/**
 * Razred BankovnaUsluga
 */
public abstract class BankovnaUsluga implements IRed {

    private int brojURedu = 0;

    public static final int MAKS_RED = 10;

    public abstract void uciniUslugu (int [] parametri );

    public boolean dodajZadnjeg()
    {
        if (brojURedu == BankovnaUsluga.MAKS_RED) return false;
        brojURedu = brojURedu + 1;
    }
}
```

```
    Ispis("novi dosao, sad ih je "+brojURedu); //metoda za ispis na konzolu
    return true;
}

public boolean ukloniPrvog()
{
    if (brojURedu == 0) return false;
    brojURedu = brojURedu - 1;
    Ispis("prvi otisao, ostaje ih "+brojURedu); //metoda za ispis na konzolu
    return true;
}
...
}
```

a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:

```
/**
 * Sučelje IRed
 */
public interface IRed {

    public boolean dodajZadnjeg ();

    public boolean ukloniPrvog ();

}
```

`IRed red = new BankovnaUsluga();`

a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:

```
IRed red = new BankovnaUsluga();
```

Rj. Nije ispravan, budući da se ne može instancirati primjerak apstraktnog razreda `BankovnaUsluga`.

6. (2 boda) Razred BankovnaUsluga i sučelje IRed opisani su sljedećim kôdom:

```
/**
 * Razred BankovnaUsluga
 */
public abstract class BankovnaUsluga implements IRed {

    private int brojURedu = 0;

    public static final int MAKS_RED = 10;

    public abstract void uciniUslugu (int [] parametri );

    public boolean dodajZadnjeg()
    {
        if (brojURedu == BankovnaUsluga.MAKS_RED) return false;
        brojURedu = brojURedu + 1;
    }
}
```

```
        Ispis("novi dosao, sad ih je "+brojURedu); //metoda za ispis na konzolu
        return true;
    }

    public boolean ukloniPrvog()
    {
        if (brojURedu == 0) return false;
        brojURedu = brojURedu - 1;
        Ispis("prvi otisao, ostaje ih "+brojURedu); //metoda za ispis na konzolu
        return true;
    }
    ...
}
```

```
/**
 * Sučelje IRed
 */
public interface IRed {

    public boolean dodajZadnjeg ();

    public boolean ukloniPrvog ();

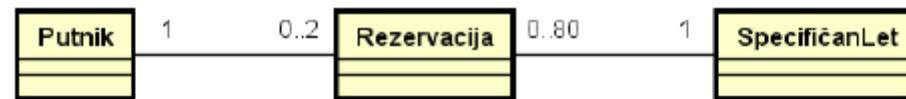
}
```

b) (1 bod) Implementira li ispravno razred BankovnaUsluga sučelje IRed? Ukratko objasnite odgovor.

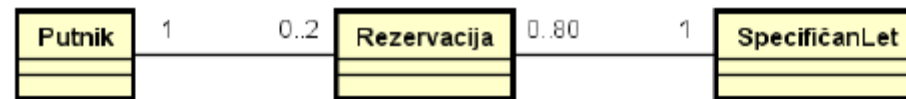
b) (1 bod) Implementira li ispravno razred `BankovnaUsluga` sučelje `IRed`? Ukratko objasnite odgovor.

Rj. Da, implementacija sučelja je ispravna, budući da za obje metode definirane u sučelju postoji valjana implementacija.

Prema donjem dijagramu razreda, odgovorite: koliki je najmanji, a koliki je najveći broj rezervacija prisutnih u sustavu, ako u sustavu postoje 5 putnika i 2 specifična leta?



Prema donjem dijagramu razreda, odgovorite: koliki je najmanji, a koliki je najveći broj rezervacija prisutnih u sustavu, ako u sustavu postoje 5 putnika i 2 specifična leta?



Rj. Svaki od 5 putnika može imati između 0 i 2 rezervacije. Najmanji broj rezervacija u sustavu je stoga 0, a najveći 10.

Obrazloženje (studenti ne moraju navoditi): Budući da je svaka rezervacija povezana s točno jednim specifičnim letom, a svaki specifični let može imati od 0 do 80 rezervacija, to se može dogoditi da sve rezervacije budu na istom letu. Ograničenje je na strani Putnik - Rezervacija, budući da 2 specifična leta mogu imati između 0 i 160 rezervacija. Međutim, ako postoji više od 160 rezervacija u sustavu, broj putnika u sustavu bi morao biti veći, što ovdje nije slučaj.

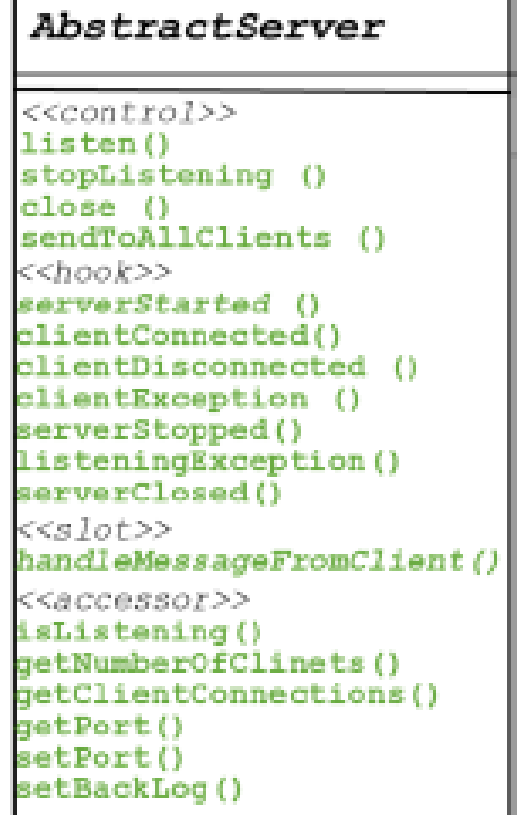
Opišite uvjete kretanja znački (engl. token) u UML dijagramu aktivnosti (engl. activity diagram).

Opišite uvjete kretanja znački (engl. token) u UML dijagramu aktivnosti (engl. activity diagram).

Rj. dio semantike bez grafičkog prikaza. Značka može predstavljati: upravljački tijek; objekt; podatak. Kreću se od izvorišta prema odredištu vezama ovisno o: ispunjenim uvjetima izvornog čvora, postavljenim uvjetima veza (engl. Edge guard conditions), preduvjetima ciljnog čvora.



UML diagramom razreda prikazan je apstraktni razred poslužitelja s navedenim stereotipima metoda u radnom okviru klijentsko-poslužiteljske arhitekture. Koje metode programer može ali i ne mora implementirati? Čemu služi ta vrsta metoda?



UML dijagramom razreda prikazan je apstraktni razred poslužitelja s navedenim stereotipima metoda u radnom okviru klijentsko-poslužiteljske arhitekture. Koje metode programer može ali i ne mora implementirati? Čemu služi ta vrsta metoda?

Sve hook metode, ima ih sedam:  
serverStarted(), clientConnected(),  
clientDisconnected(), clientException(),  
serverStopped(), listeningException() i  
serverClosed().

To su metode koje se mogu po potrebi redefinirati. U radnom okviru za njih postoji minimalna implementacija.

```
AbstractServer
<<control>>
listen()
stopListening ()
close ()
sendToAllClients ()
<<hook>>
serverStarted ()
clientConnected()
clientDisconnected ()
clientException ()
serverStopped()
listeningException()
serverClosed()
<<slot>>
handleMessageFromClient()
<<accessor>>
isListening()
getNumberOfClients ()
getClientConnections()
getPort ()
setPort ()
setBackLog ()
```

Koji je cilj provođenja aktivnosti  
ispitivanja u procesu oblikovanja  
programske potpore?

# Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

Rješenje: Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema ispitivane programske podrške.

Priznaje se i kraće rješenje: pronalaženje pogrešaka.

12. (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*):
- a) (1 bod) Navedite korake ispitivanja.
  - b) (1 bod) Na primjeru jednog ulaznog 4-znamenkastog broja u intervalu [5000, 9998], navedite potrebne minimalne ispitne slučajeve.

1. Odredi particije za sve ulazne varijable.
2. Za sve particije odaberi vrijednosti ispitivanja.
3. Definiraj ispitne slučajeve koristeći odabrane vrijednosti.
4. Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.

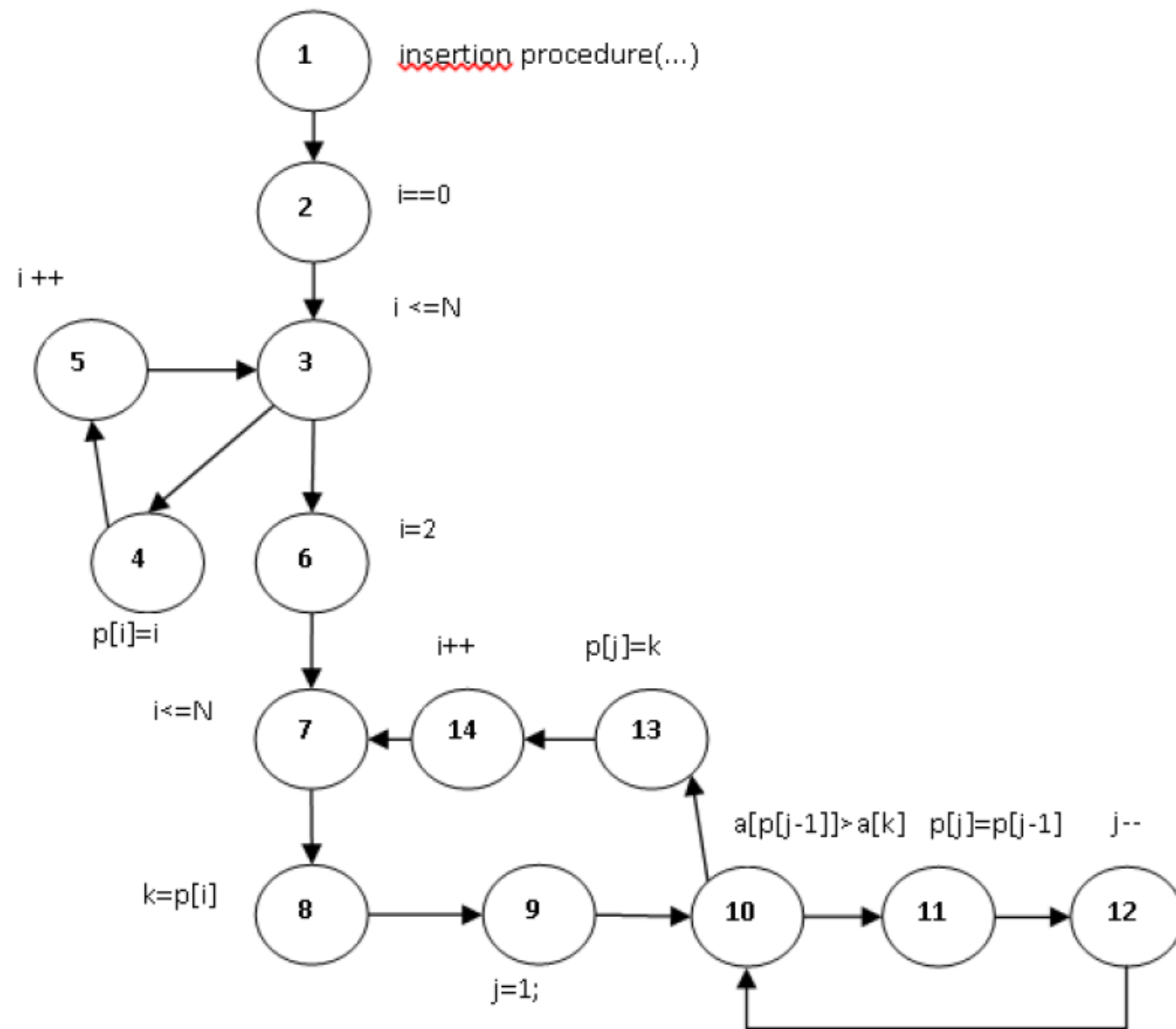
Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	4999	ne prolazi
IS2	5000	prolazi
IS3	8001	prolazi
IS4	9998	prolazi
IS5	9999	ne prolazi

\*priznaju se i dodatni ispitni slučajevi: (4000, ne prolazi)

13. (2 boda) Za sljedeću funkciju:

- a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.
- b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

```
01. insertion_procedure (int a[], int p [], int N)
02. {
03.     int i,j,k;
04.     for (i=0;i<=N; i++)
05.         p[i] = i;
06.     for (i=2;i<=N; i++)
07.     {
08.         k=p[i];j=1;
09.         while (a[p[j-1]] > a[k]) {
10.             p[j] = p[j-1];
11.             j--
12.         }
13.         p[j] = k;
14.     }
```



$$CV(G) = \text{Lukovi} - \text{Čvorovi} + 2 * P = 16 - 14 + 2 = 4$$



Navedite osnovno svojstvo razina u općenitoj višerazinskoj arhitekturi (engl. n-tier architecture).

Navedite osnovno svojstvo razina u općenitoj višerazinskoj arhitekturi (engl. n-tier architecture).

Rj. Predstavlja proširenje raspodijeljene arhitekture klijent – poslužitelj. Razina zatvara (skriva, enkapsulira) skup usluga i implementacijske detalje niže razine o kojoj ovisi.

Što određuje tijek izvođenja programa u arhitekturi protoka podataka (engl. data flow)?

# Što određuje tijek izvođenja programa u arhitekturi protoka podataka (engl. data flow)?

Rj. Redoslijed izvršavanja instrukcija nije određen programskim brojiлом već već je nedeterminističan, odnosno ovisan o podacima koji su raspoloživi aktorima (engl. functional unit - FUs). Pomakom podataka aktivira se daljnje upravljanje.

U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. version control, revision control, source control). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. version control, revision control, source control). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

Rj. u lokalnim repozitorijima kod članova tima, moguć je istodoban rad.

Definirajte što znači da je skup  
logičkih formula  $\Gamma$  konzistentan.

# Definirajte što znači da je skup logičkih formula $\Gamma$ konzistentan.

Rj. Skup  $\Gamma$  je konzistentan ako ne sadrži formule na temelju kojih bi formule  $\omega_i$  i  $\neg\omega_i$  (istovremeno) bili teoremi (dedukcije).



Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda: "Na nekom fakultetu postoji samo jedan izborni predmet 'Ekspertni sustavi'."

Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda: "Na nekom fakultetu postoji samo jedan izborni predmet 'Ekspertni sustavi'."

Rj.

$F(x)$  = x je fakultet

$IP(x,y)$  = x je izborni predmet na y

$=(x,y)$  = x je jednako y

'Ekspertni sustavi' = konst.

$\exists x (F(x) \wedge IP('Ekspertni sustavi', x) \wedge \neg \exists y (IP(y,x) \wedge \neg =( 'Ekspertni sustavi', y)))$

alternativno:

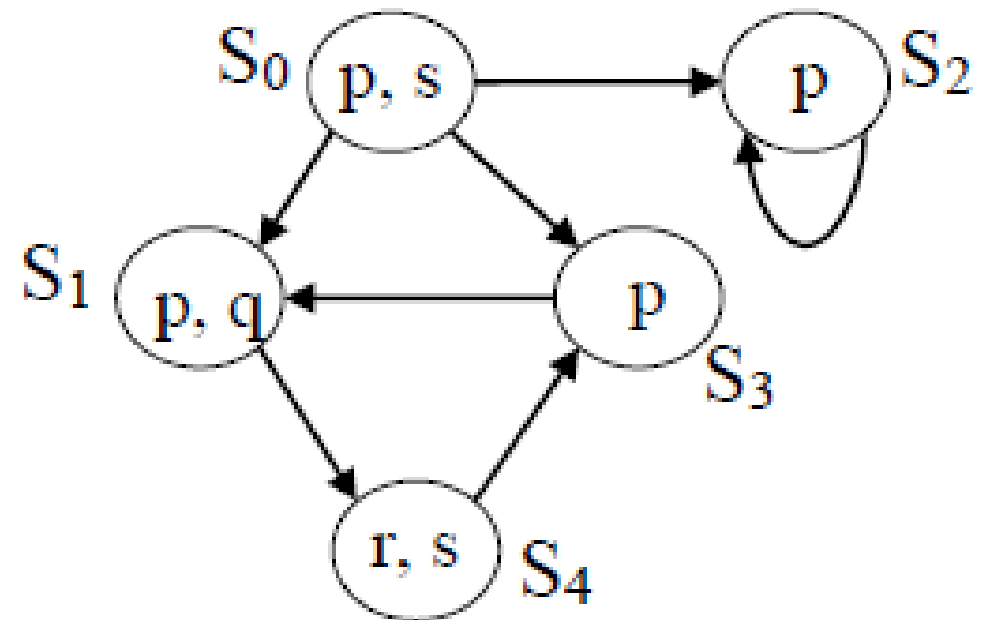
$\exists x (F(x) \wedge IP('Ekspertni sustavi', x) \wedge \forall y (IP(y,x) \Rightarrow =( 'Ekspertni sustavi', y)))$

Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic): "Uvijek nakon  $req=1$  konačno dođe  $ack=1$ ."

Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic): "Uvijek nakon  $\text{req}=1$  konačno dođe  $\text{ack}=1$ ."

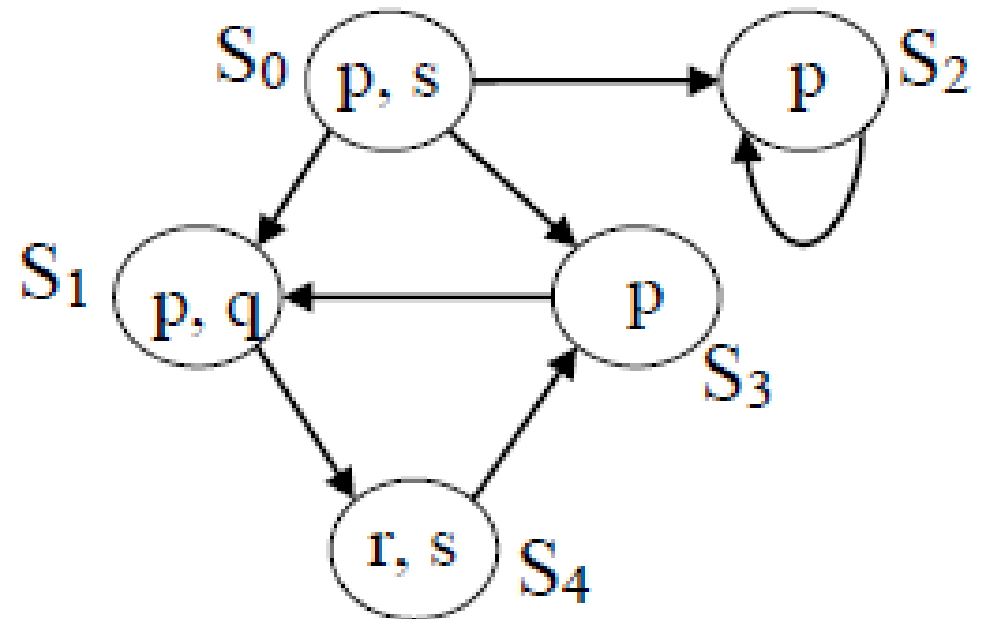
Rj.  $\text{AG} (\text{req}=1 \Rightarrow \text{AF ack}=1)$ .

Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu  $A$  ( $p \cup r$ ).



Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu  $A \ (p \cup r)$ .

Rj. Takva stanja  
 $su = S1, S3, S4$



OPP\_ZI\_2016\_2017\_grupaA

Navedite na koje se sve načine  
mogu izraziti zahtjevi sustava.



# Navedite na koje se sve načine mogu izraziti zahtjevi sustava.

Rj. strukturiranim prirodnim jezikom, specijalnim jezikom za opis oblikovanja (npr. SDL), grafičkom notacijom (npr. UML) i matematičkom specifikacijom (FSM, teorija skupova, logika).

Navedite generičke aktivnosti  
inženjerstva zahtjeva.

Navedite generičke aktivnosti inženjerstva zahtjeva.

Rj. studija izvedivosti, izlučivanje zahtjeva , analiza i specifikacija zahtjeva, validacija zahtjeva, upravljanje zahtjevima

Navedite barem dvije značajke metodologije ubrzanog razvoja (engl. agile methodology) programske potpore.

Navedite barem dvije značajke metodologije ubrzanog razvoja (engl. agile methodology) programske potpore.

Rj. Iterativni razvoj, mali inkrementi, kontinuirano poboljšanje PP, naglasak na ljude i suradnju, uključenost korisnika u proces razvoja...

Koja je temeljna značajka  
vodopadnog modela (engl.  
Waterfall model)  
razvoja programske potpore?

Koja je temeljna značajka  
vodopadnog modela (engl.  
Waterfall model)  
razvoja programske potpore?

Rj. Prethodna faza treba se  
završiti prije prelaska na novu  
fazu.

Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili korisničke zahtjeve?



Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili korisničke zahtjeve?

UCovi s pripadajućim dijagramima  
(funkcionalni zahtjevi), sekv.  
dijagrami, lista  
nefunkcionalnih zahtjeva

6. **(2 boda)** Razredi Lik2D te razredi Kvadrat i Pravokutnik, koji ga nasljeđuju, opisani su sljedećim kodom:

```
/**
 * Razred Lik2D
 */
public abstract class Lik2D {

    public abstract int Povrsina (int parametri[] );

    public void IdentificirajSe()
    {
        Ispis("Ja sam Lik 2D"); //metoda za ispis na konzolu
    }
}

/**
 * Razred Kvadrat nasljedjuje Lik2D
 */
```

```
public class Kvadrat extends Lik2D{

    public int Povrsina(int parametri[]){
        return parametri[0]*parametri[0];
    }

    public void IdentificirajSe(){
        Ispis ("Ja sam Kvadrat");
    }
}

/**
 * Razred Pravokutnik nasljedjuje Lik2D
 */
public class Pravokutnik extends Lik2D{

    public int Povrsina (int parametri[]){
        return parametri[0]*parametri[1];
    }
}
```

- a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:

Kvadrat k = new Lik2D();

a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:

```
Kvadrat k = new Lik2D();
```

- Rj. Nije ispravan, budući da se ne može instancirati primjerak apstraktnog razreda `Lik2D`.

6. (2 boda) Razredi Lik2D te razredi Kvadrat i Pravokutnik, koji ga nasljeđuju, opisani su sljedećim kodom:

```
/**
 * Razred Lik2D
 */
public abstract class Lik2D {

    public abstract int Povrsina (int parametri[] );

    public void IdentificirajSe()
    {
        Ispis("Ja sam Lik 2D"); //metoda za ispis na konzolu
    }
}

/**
 * Razred Kvadrat nasljedjuje Lik2D
 */
```

---

Oblikovanje programske potpore

```
public class Kvadrat extends Lik2D{

    public int Povrsina(int parametri[]){
        return parametri[0]*parametri[0];
    }

    public void IdentificirajSe(){
        Ispis ("Ja sam Kvadrat");
    }
}

/**
 * Razred Pravokutnik nasljedjuje Lik2D
 */
public class Pravokutnik extends Lik2D{

    public int Povrsina (int parametri[]){
        return parametri[0]*parametri[1];
    }
}
```

- b) (1 bod) Što će se ispisati u konzoli prilikom izvođenja metode main() čiji se kod nalazi u nastavku zadatka?

```
public static void main(String[] args) {

    Lik2D k = new Kvadrat();
    Lik2D p = new Pravokutnik();

    int []parametri_k = new int[1];
    int []parametri_p = new int[2];

    parametri_k[0]=5;
    parametri_p[0]=2;
    parametri_p[1]=3;

    Ispis(k.Povrsina(parametri_k));
    Ispis(p.Povrsina(parametri_p));

    k.IdentificirajSe();
    p.IdentificirajSe();
}
```

```

public static void main(String[] args) {

    Lik2D k = new Kvadrat();
    Lik2D p = new Pravokutnik();

    int []parametri_k = new int[1];
    int []parametri_p = new int[2];

    parametri_k[0]=5;
    parametri_p[0]=2;
    parametri_p[1]=3;

    Ispis(k.Povrsina(parametri_k));
    Ispis(p.Povrsina(parametri_p));

    k.IdentificirajSe();
    p.IdentificirajSe();
}

```

- Rj.
- 25
- 6
- Ja sam Kvadrat
- Ja sam Lik2D

Prema donjem dijagramu razreda, odgovorite: koliko najviše putnika može svako dizalo podići te mora li putnik uopće putovati dizalom?



Prema donjem dijagramu razreda, odgovorite: koliko najviše putnika može svako dizalo podići te mora li putnik uopće putovati dizalom?



Rj. Svako dizalo može podići najviše 8 putnika, a putnik ne mora putovati dizalom.

Navedite sve temeljne elemente  
sekvencijskog dijagrama i označite  
ih na  
skiciranom hipotetskom primjeru.



Navedite sve temeljne elemente  
sekvencijskog dijagrama i označite  
ih na  
skiciranom hipotetskom primjeru.

Rj. Aktori (ili objekti), životne linije,  
aktivnosti, poruke (može i petlje, grananja,  
sinkrone/asinkrone/create/destroy  
poruke)  
<<neka skica ovdje>>...

Kako je organiziran primjenski  
program u arhitekturi zasnovanoj  
na  
uslugama?

Kako je organiziran primjenski program u arhitekturi zasnovanoj na uslugama?

Rj. Uslužno usmjerena arhitektura organizira primjenski program (cjelovitu aplikaciju) kao kolekciju usluga koje međusobno komuniciraju uporabom dobro definiranih sučelja.

Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)? Naputak: rješenje prikažite tablično.

Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)? Naputak: rješenje prikažite tablično.

<i>Naziv</i>	<i>Razumljivo ime</i>
<i>Opis</i>	<i>Opis problema</i>
<i>Rješenje</i>	<i>Opis predložka koji može biti upotrijebljen na različite načine</i>
<i>Posljedice</i>	<i>Rezultati i pogodnosti primjene obrasca uporabe</i>

Navedite kojoj skupini (ili kojim sve skupinama) označenom stereotipom pripadaju metode u razredu AbstractClient arhitekture OCSF koje se ne mogu redefinirati u podrazredima.



Navedite kojoj skupini (ili kojim sve skupinama) označenom stereotipom pripadaju metode u razredu AbstractClient arhitekture OCSF koje se ne mogu redefinirati u podrazredima.

Rješenje: <<control>> i <<accessor>>



Za slučaj arhitekture klijent-poslužitelj, kada postoji n spojenih klijenata,  
izračunajte minimalan broj dretvi pri radu poslužitelja implementiranog objektno usmjerenim radnim okvirom OCSF (NAPOMENA: zanemarite administracijske dretve, dretve OS-a, VM,...).



Za slučaj arhitekture klijent-poslužitelj, kada postoji  $n$  spojenih klijenata,  
izračunajte minimalan broj dretvi pri radu poslužitelja implementiranog objektno usmjerenim radnim okvirom OCSF (NAPOMENA: zanemarite administracijske dretve, dretve OS-a, VM,...).

**Rješenje:  $n+1$**

Program koji kao ulaze prima tri dvoznamenkasta cijela broja unesena tipkovnicom potrebno je ispitati primjenom tehnike kombinacijskog ispitivanja (engl. Combination testing). Izračunajte broj potrebnih ispitnih slučajeva.

Program koji kao ulaze prima tri dvoznamenkasta cijela broja unesena tipkovnicom potrebno je ispitati primjenom tehnike kombinacijskog ispitivanja (engl. Combination testing). Izračunajte broj potrebnih ispitnih slučajeva.

Rješenje: ulazne vrijednosti -99..99

- $199 * 199 * 199 = 7\,880\,599$
- bilo bi izvrsno da studenti prokomentiraju i neispravne vrijednosti ulaza.

Objasnite postupak inkrementalnog integracijskog ispitivanja odozgo na dolje (engl. Top down integration)

# Objasnite postupak inkrementalnog integracijskog ispitivanja odozgo na dolje (engl. Top down integration)

Razviti kostur sustava i postepeno ga popuniti komponentama, uz inkrementalno ispitivanje

- Nije potrebno razvijati upravljačke programe, ali treba razvijati prividne komponente

16.(2 boda) Za sljedeću funkciju:

- a) (1 bod) nacrtajte graf tijeka programa,
- b) (1 bod) odredite njezinu ciklomatsku složenost. Navedite formulu za izračun.

```
public void bubbleSort(int [] array) {  
    int c, d, swap, n;  
    n = array.length;  
    for (c = 0 ; c < ( n - 1 ); c++)  
    {  
        for (d = 0 ; d < n - c - 1; d++)  
        {  
            if (array[d] > array[d+1])  
            {  
                swap      = array[d];  
                array[d]   = array[d+1];  
                array[d+1] = swap;  
            }  
        }  
    }  
    return;  
}
```

16.(2 boda) Za sljedeću funkciju:

- a) (1 bod) nacrtajte graf tijeka programa,
- b) (1 bod) odredite njezinu ciklomatsku složenost. Navedite formulu za izračun.

Rj.

a)

```
int c, n, swap;  
n = array.length;    (1)  
for (c = 0 (2); c < ( n - 1 ) (3); c++ (6))  
{  
    for (d = 0 (4); d < n - c - 1 (5); d++ (11))  
    {  
        if (array[d] > array[d+1]) (7)  
        {  
            swap      = array[d];    (8)  
            array[d]   = array[d+1];  (9)  
            array[d+1] = swap;        (10)  
        }  
    }  
}  
return; (12)
```

16.(2 boda) Za sljedeću funkciju:

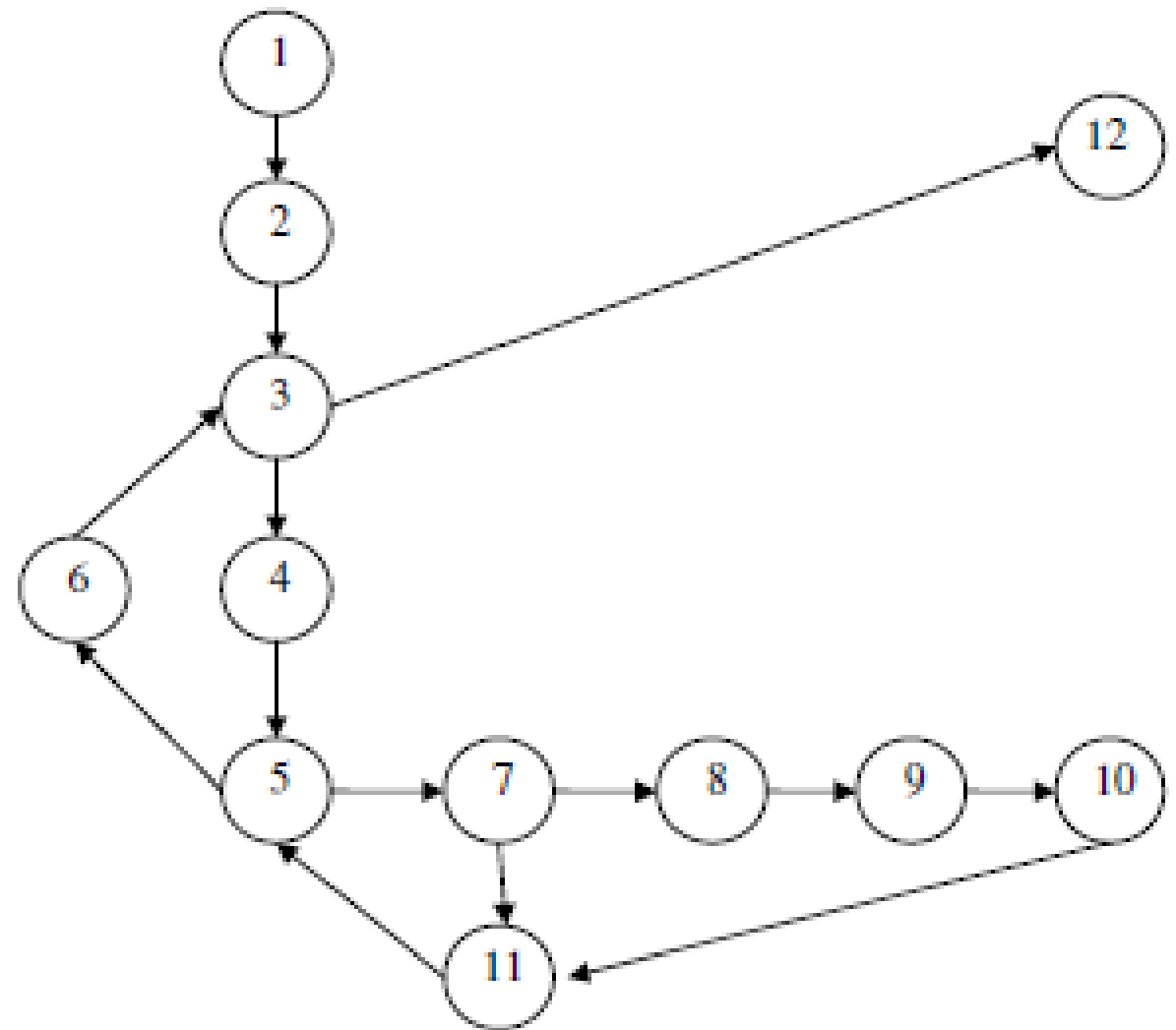
- (1 bod) nacrtajte graf tijeka programa,
- (1 bod) odredite njezinu ciklomatsku složenost. Navedite formulu za izračun.

```
public void bubbleSort(int [] array) {  
    int c, d, swap, n;  
    n = array.length;  
    for (c = 0 ; c < ( n - 1 ); c++)  
    {  
        for (d = 0 ; d < n - c - 1; d++)  
        {  
            if (array[d] > array[d+1])  
            {  
                swap      = array[d];  
                array[d]  = array[d+1];  
                array[d+1] = swap;  
            }  
        }  
    }  
    return;  
}
```

KJ.

a)

```
int c, n, swap;  
n = array.length; (1)  
for (c = 0 (2); c < ( n - 1 ) (3); c++ (6))  
{  
    for (d = 0 (4); d < n - c - 1 (5); d++ (11))  
    {  
        if (array[d] > array[d+1]) (7)  
        {  
            swap      = array[d]; (8)  
            array[d]  = array[d+1]; (9)  
            array[d+1] = swap; (10)  
        }  
    }  
}
```



b)  $CV(G) = 14 - 12 + 2 = 4$



U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. version control, revision control, source control). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. version control, revision control, source control). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

Rj. u lokalnim repozitorijima kod članova tima, moguć je istodoban rad.

Definirajte potrebne predikate i konstante  
te preslikajte rečenicu u dobro  
definiranu formulu predikatne logike  
prvoga reda:

"Dora proučava samo jednu veliku knjigu."

Definirajte potrebne predikate i konstante  
te preslikajte rečenicu u dobro  
definiranu formulu predikatne logike  
prvoga reda:

"Dora proučava samo jednu veliku knjigu."

Rj.

$VK(x)$  = x je velika knjiga

$P(x,y)$  = x proučava y

$=(x,y)$  = x je jednako y

Dora = konst.

$\exists x (VK(x) \wedge P(Dora,x) \wedge \neg \exists y (VK(y) \wedge P(Dora,y) \wedge \neg =(x,y)))$

alternativno:

$\exists x (VK(x) \wedge P(Dora,x) \wedge \forall y ((VK(y) \wedge P(Dora,y)) \Rightarrow =(x,y)))$

Što mora biti zadovoljeno za interpretacije skupa formula  $G$ , tako da možemo tvrditi da vrijedi  $G \models P$  tj. da je formula  $P$  logička posljedica skupa  $G$ ?

Što mora biti zadovoljeno za interpretacije skupa formula  $G$ , tako da možemo tvrditi da vrijedi  $G \models P$  tj. da je formula  $P$  logička posljedica skupa  $G$ ?

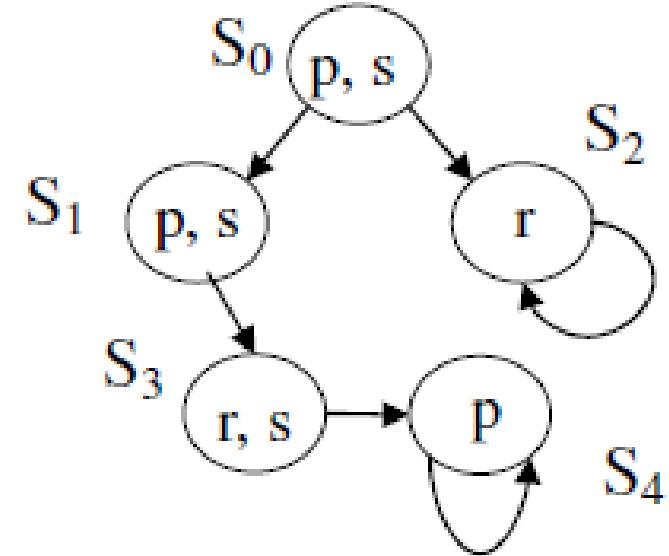
Rj. Svaka interpretacija koja za skup  $G$  daje istinitost mora i za  $P$  dati istinitost.

Za zadani model implementacije Kripke strukturom  $M$  prema slici

potrebno je:

a) Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).

b) Odrediti sva stanja koja zadovoljavaju formulu  $A (s \cup p)$ .



Za zadani model implementacije Kripke strukturom M prema slici

potrebno je:

- Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).
- Odrediti sva stanja koja zadovoljavaju formulu  $A (s \cup p)$ .

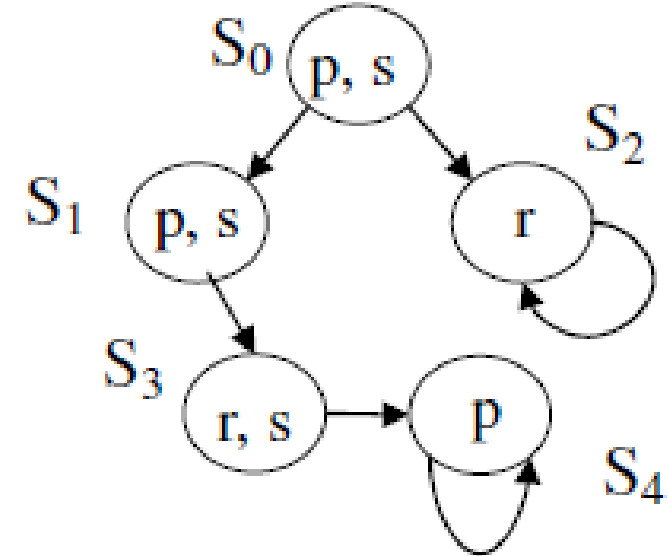
Rj.

a)  $S = \{S_0, S_1, S_2, S_3, S_4\}$

$R = \{(S_0, S_1), (S_0, S_2), (S_2, S_2), (S_1, S_3), (S_3, S_4), (S_4, S_4)\}$ .

$L(S_0) = \{p, s\}$ ;  $L(S_1) = \{p, s\}$ ;  $L(S_2) = \{r\}$ ;  $L(S_3) = \{r, s\}$ ;  
 $L(S_4) = \{p\}$

b)  $S_0$  - da,  $S_1$  - da,  $S_2$  - ne,  $S_3$  - da,  $S_4$  - da.





ZI 16/17 B

- Što su zahtjevi domene primjene?

- Što su zahtjevi domene primjene?
- Rj. Funkc. i nefunkc. zahtjevi koji proizlaze iz domene primjene/specifični su za domenu
- primjene/karakteriziraju domenu primjene.

- Navedite generičke aktivnosti procesa programskog inženjerstva.

- Navedite generičke aktivnosti procesa programskog inženjerstva.
- Rj. Specifikacija, oblikovanje i implementacija, validacija i verifikacija, evolucija

- Navedite barem dvije značajke metodologije ubrzanog razvoja (engl.
- agile methodology) programske potpore.

- Navedite barem dvije značajke metodologije ubrzanog razvoja (engl. agile methodology) programske potpore.
- ☐ Rj. Iterativni razvoj, mali inkrementi, kontinuirano poboljšanje PP, naglasak na ljude i suradnju, uključenost korisnika u proces razvoja...

- 4. (1 bod) Koja je temeljna značajka vodopadnog modela (engl. Waterfall model)
- razvoja programske potpore?



- 4. (1 bod) Koja je temeljna značajka vodopadnog modela (engl. Waterfall model)
- Razvoja programske potpore?
- Rj. Prethodna faza treba se završiti prije prelaska na novu fazu.

- Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili
- korisničke zahtjeve?

- Navedite na koje ste sve načine u projektnoj dokumentaciji izrazili
- korisničke zahtjeve?
- ☐ Rj. UCovi s pripadajućim dijagramima (funkcionalni zahtjevi), sekv. dijagrami, lista
- nefunkcionalnih zahtjeva

a) Pas z = new Zivotinja(); b)

6. (2 boda) Razredi Zivotinja te razredi Pas i Macka koji ga nasljeđuju, opisani su sljedećim kodom:

```
/**
 * Razred Zivotinja
 */
public abstract class Zivotinja {

    public abstract String oglasiSe (int parametar);

    public void IdentificirajSe()
    {
        Ispis("Ja sam Zivotinja"); //metoda za ispis na
    }
}

/**
 * Razred Pas nasljedjuje Zivotinja
 */
public class Pas extends Zivotinja{

    public String oglasiSe (int parametar){
        if (parametar == 0) return "Vau!";
        else return "Grrr!";
    }

    public void IdentificirajSe(){
        Ispis ("Ja sam Pas");
    }
}

/**
 * Razred Macka nasljedjuje Zivotinja
 */
public class Macka extends Zivotinja{

    public String oglasiSe (int parametar){
        if (parametar == 0) return "Mijau";
        else return "Mrrrnjau!";
    }
}
```

a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi main():

- a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:
- `Pas z = new Zivotinja();`
- ☐ Rj. Nije ispravan, budući da se ne može instancirati primjerak apstraktnog razreda
- Zivotinja.

b) (1 bod) Što će se ispisati u konzoli prilikom izvođenja metode `main()` čiji se kod nalazi u nastavku zadatka?

```
public static void main(String[] args) {  
  
    Zivotinja p = new Pas();  
    Zivotinja m = new Macka();  
  
    int parametar_p = 0;  
    int parametar_m = 1;  
  
    Ispis(p.oglasise(parametar_m));  
    Ispis(m.oglasise(parametar_p));  
  
    p.IdenticirajSe();  
    m.IdenticirajSe();  
}
```

- ☐ Rj.
- ☐ Grrr!
- ☐ Mijau!
- ☐ Ja sam Pas.
- ☐ Ja sam Zivotinja.

7. **(1 bod)** Prema donjem dijagramu razreda, odgovorite: koliko se najmanje računala mora nalaziti u laboratoriju te mora li se svako računalo nalaziti u nekom laboratoriju?





- Rj. Najmanje 4 računala moraju se nalaziti u laboratoriju i svako se računalo mora nalaziti u nekom laboratoriju.

7. **(1 bod)** Prema donjem dijagramu razreda, odgovorite: koliko se najmanje računala mora nalaziti u laboratoriju te mora li se svako računalo nalaziti u nekom laboratoriju?



- Navedite sve temeljne elemente komunikacijskog dijagrama i označite ih
- na skiciranom hipotetskom primjeru.

- Navedite sve temeljne elemente komunikacijskog dijagrama i označite ih
- na skiciranom hipotetskom primjeru.
- Rj. Aktori (ili objekti), veze između aktora/objekata, poruke sa zadanim smjerom i brojčanim
- redoslijedom izvođenja.
- <<neka skica ovdje>>...

- Kako je organiziran primjenski program u arhitekturi zasnovanoj na
- uslugama?

- Kako je organiziran primjenski program u arhitekturi zasnovanoj na
- uslugama?
- ☐ Rj. Uslužno usmjerena arhitektura organizira primjenski program (cjelovitu aplikaciju) kao
- kolekciju usluga koje međusobno komuniciraju uporabom dobro definiranih sučelja.

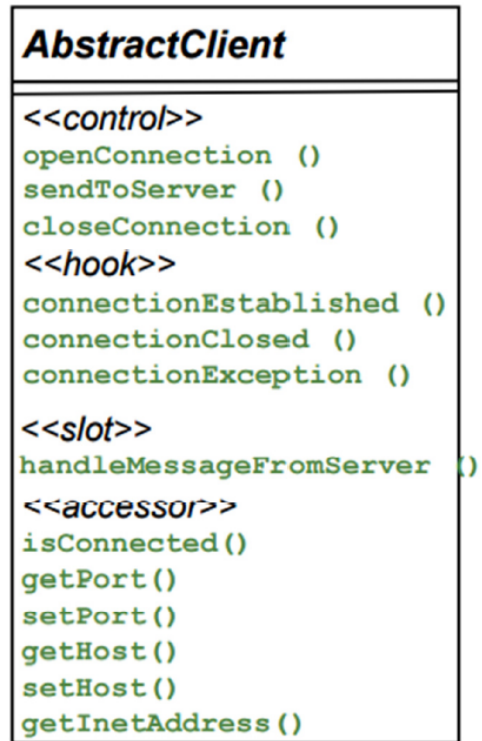
- Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)? Naputak: rješenje prikazite tablično.

- Koji su minimalni elementi kojima se opisuju oblikovni obrasci (engl. design pattern)? Naputak: rješenje prikažite tablično.

<i>Naziv</i>	<i>Razumljivo ime</i>
<i>Opis</i>	<i>Opis problema</i>
<i>Rješenje</i>	<i>Opis predložka koji može biti upotrijebljen na različite načine</i>

<i>Posljedice</i>	<i>Rezultati i pogodnosti primjene obrasca uporabe</i>
-------------------	--

12. **(1 bod)** Navedite kojoj skupini (ili kojim sve skupinama) označenom stereotipom pripadaju metode u razredu AbstractClient arhitekture OCSF koje se ne mogu redefinirati u podrazredima.





- Controler i accessor

- Za slučaj arhitekture klijent-poslužitelj, kada postoji n spojenih klijenata,
- izračunajte minimalan broj dretvi pri radu poslužitelja implementiranog objektno
- usmjerenim radnim okvirom OCSF (NAPOMENA: zanemarite administracijske
- dretve, dretve OS-a, VM,...).

- Rješenje:  $n+1$

- Funkcija ima tri parametra koji su dvoznamenkasti cijeli brojevi. Funkciju
- je potrebno ispitati primjenom tehnike kombinacijskog ispitivanja (engl.
- Combination testing). Izračunajte broj potrebnih ispitnih slučajeva.

- Funkcija ima tri parametra koji su dvoznamenkasti cijeli brojevi. Funkciju
- je potrebno ispitati primjenom tehnike kombinacijskog ispitivanja (engl. Combination testing). Izračunajte broj potrebnih ispitnih slučajeva.
- ☐ Rješenje: ulazne vrijednosti -99..99
- ☐  $199 * 199 * 199 = 7\,880\,599$
- ☐ bilo bi izvrsno da studenti prokomentiraju i neispravne vrijednosti ulaza.

- Objasnite postupak inkrementalnog integracijskog ispitivanja odozdo na
- gore (engl. Bottom Up integration)

- Objasnite postupak inkrementalnog integracijskog ispitivanja odozdo na
- gore (engl. Bottom Up integration)
- ☐ Razvijaju se i ispituju najprije krajnje komponente, a program se izgrađuje inkrementalno
- povezuju ove komponente u veće cjeline.
- ☐ Potrebno je razvijati upravljačke programe, ali nisu potrebne prividne komponente.

- Za sljedeću funkciju:
- a) (1 bod) nacrtajte graf tijeka programa,
- b) (1 bod) odredite njezinu ciklomatsku složenost. Navedite formulu za izračun.

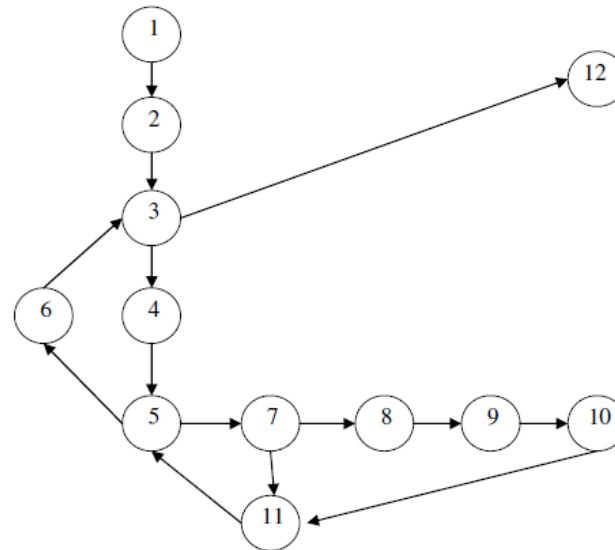
```
public void bubbleSort(int [] array) {  
    int c, d, swap, n;  
    n = array.length;  
    for (c = 0 ; c < ( n - 1 ); c++)  
    {  
        for (d = 0 ; d < n - c - 1; d++)  
        {  
            if (array[d] > array[d+1])  
            {  
                swap      = array[d];  
                array[d]   = array[d+1];  
                array[d+1] = swap;  
            }  
        }  
    }  
    return;  
}
```



b)  $CV(G) = 14 - 12 + 2 = 4$

a)

```
int c, n, swap;  
n = array.length; (1)  
for (c = 0 (2); c < ( n - 1 ) (3); c++ (6))  
{  
  for (d = 0 (4); d < n - c - 1 (5); d++ (11))  
  {  
    if (array[d] > array[d+1]) (7)  
    {  
      swap      = array[d]; (8)  
      array[d]  = array[d+1]; (9)  
      array[d+1] = swap; (10)  
    }  
  }  
}  
return; (12)
```



- 17. (1 bod) U timskom razvoju programske potpore pojedine dijelove istodobno
- razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama
- datoteka (engl. Version control, revision control, source control). Gdje su
- smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja
- (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na istom
- izvornom kodu?

- 17. (1 bod) U timskom razvoju programske potpore pojedine dijelove istodobno
- razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama
- datoteka (engl. Version control, revision control, source control). Gdje su
- smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja
- (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na istom
- izvornom kodu?
- ☐ Rj. u lokalnim repozitorijima kod članova tima, moguć je istodoban rad.

- 18. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro
- definiranu formulu predikatne logike prvoga reda:
- "Ana promijeni svaki recept za tortu."

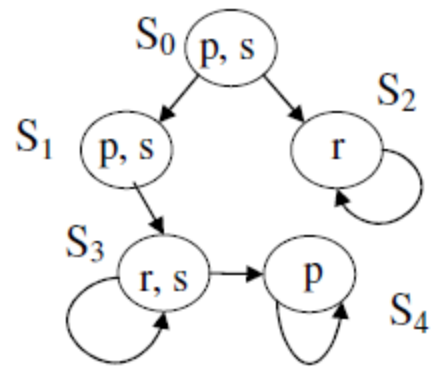
- 18. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:
- "Ana promijeni svaki recept za tortu."
- $R_j$ .
- $P(x,y)$  = x promijeni y
- $RT(x)$  = x je recept za tortu
- Ana = konst.
- $\forall x (RT(x) \rightarrow P(\text{Ana},x))$

- 19.(1 bod) Što je to model formalnog logičkog sustava?

- 19.(1 bod) Što je to model formalnog logičkog sustava?
- Rj. Neka interpretacija je model FLS ako evaluiira sve njegove formule u istinito.

20. **(2 boda)** Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je:

- Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).
- Odrediti sva stanja koja zadovoljavaju formulu  $A$  ( $s \cup p$ ).





Rj.

a)  $S = \{S_0, S_1, S_2, S_3, S_4\}$

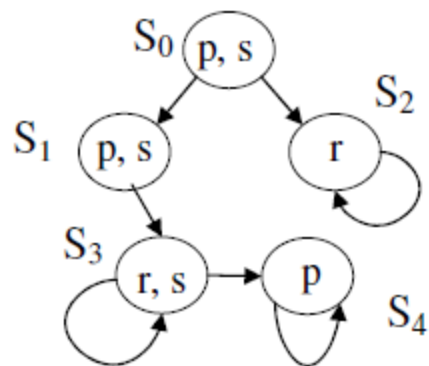
$R = \{(S_0, S_1), (S_0, S_2), (S_2, S_2), (S_1, S_3), (S_3, S_3), (S_3, S_4), (S_4, S_4)\}$ .

$L(S_0) = \{p, s\}$ ;  $L(S_1) = \{p, s\}$ ;  $L(S_2) = \{r\}$ ;  $L(S_3) = \{r, s\}$ ;  $L(S_4) = \{p\}$

b)  $S_0$  - da,  $S_1$  - da,  $S_2$  - ne,  $S_3$  - ne,  $S_4$  - da.

20. **(2 boda)** Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je:

- a) Odrediti  $S$  (skup stanja),  $R$  (relaciju prijelaza),  $L$  (funkciju označavanja).
- b) Odrediti sva stanja koja zadovoljavaju formulu  $A$  ( $s \cup p$ ).



17/18 a

- 1. (1 bod) Navedite značajke dobrog programskog proizvoda.

- 1. (1 bod) Navedite značajke dobrog programskog proizvoda.
- Rj. Programski proizvod mora osigurati traženu funkcionalnost i performanse, te mora biti prihvatljiv korisniku, pouzdan i mora se moći održavati.

- 2. (1 bod) Navedite metode izlučivanja korisničkih zahtjeva.

- 2. (1 bod) Navedite metode izlučivanja korisničkih zahtjeva.
- Rj. intervjuiranje, scenarij, obrasci uporabe, dinamičke interakcije korištenjem sekvencijskih dijagrama.

- 3. (1 bod) Navedite tri osnovna Scrum artefakta.

- 3. (1 bod) Navedite tri osnovna Scrum artefakta.
- Rj. Projektni dnevnik zaostataka (Product Backlog), Sprint dnevnik (Sprint backlog) i inkrement.



- 4. (1 bod) Navedite klasifikaciju arhitekture programske podpore po dosegu.

- Rj. Konceptijska, logička, izvršna.

- 5. (1 bod) Objasnite princip dobrog oblikovanja programske podpore: oblikuj konzervativno (engl. design defensively).

- Rj. Ne koristiti pretpostavke kako će netko upotrebljavati oblikovanu komponentu obraditi sve slučajeve u kojima se komponenta može neprikladno upotrijebiti provjeriti valjanost ulaza u komponentu provjerom definiranih pretpostavki.

6. (2 boda) Razred BankovnaUsluga i sučelje IRed opisani su sljedećim kôdom:

```
/**
 * Razred BankovnaUsluga
 */
public abstract class BankovnaUsluga implements IRed {

    private int brojURedu = 0;

    public static final int MAKS_RED = 10;

    public abstract void uciniUslugu (int [] parametri );

    public boolean dodajZadnjeg()
    {
        if (brojURedu == BankovnaUsluga.MAKS_RED) return false;
```

---

Oblikovanje programske potpore

Stranica 1/9

```
        brojURedu = brojURedu + 1;
        Ispis("novi dosao, sad ih je "+brojURedu); //metoda za ispis na konzolu
        return true;
    }

    public boolean ukloniPrvog()
    {
        if (brojURedu == 0) return false;
        brojURedu = brojURedu - 1;
        Ispis("prvi otisao, ostaje ih "+brojURedu); //metoda za ispis na konzolu
        return true;
    }
    ...
}

/**
 * Sučelje IRed
 */
public interface IRed {

    public boolean dodajZadnjeg ();

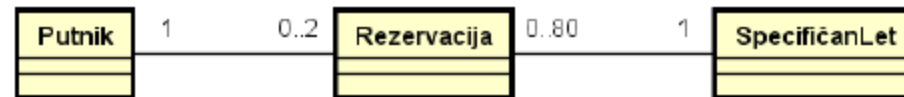
    public boolean ukloniPrvog ();

}
```

- a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:
- `IRed red = new BankovnaUsluga();`
- b) (1 bod) Implementira li ispravno razred `BankovnaUsluga` sučelje `IRed`? Ukratko objasnite odgovor.

- a) (1 bod) Obrazložite ispravnost/neispravnost sljedećeg retka u metodi `main()`:
  - `IRed red = new BankovnaUsluga();`
  - Rj. Nije ispravan, budući da se ne može instancirati primjerak apstraktnog razreda `BankovnaUsluga`.
- b) (1 bod) Implementira li ispravno razred `BankovnaUsluga` sučelje `IRed`? Ukratko objasnite odgovor.
  - Rj. Da, implementacija sučelja je ispravna, budući da za obje metode definirane u sučelju postoji valjana implementacija.

- 7. (1 bod) Prema donjem dijagramu razreda, odgovorite: koliki je najmanji, a koliki je najveći broj rezervacija prisutnih u sustavu, ako u sustavu postoje 5 putnika i 2 specifična leta?





- Rj. Svaki od 5 putnika može imati između 0 i 2 rezervacije. Najmanji broj rezervacija u sustavu je stoga **0**, a najveći **10**.
- Obrazloženje (studenti ne moraju navoditi): Budući da je svaka rezervacija povezana s točno jednim specifičnim letom, a svaki specifični let može imati od 0 do 80 rezervacija, to se može dogoditi da sve rezervacije budu na istom letu. Ograničenje je na strani Putnik - Rezervacija, budući da 2 specifična leta mogu imati između 0 i 160 rezervacija. Međutim, ako postoji više od 160 rezervacija u sustavu, broj putnika u sustavu bi morao biti veći, što ovdje nije slučaj.

- (1 bod) Opišite uvjete kretanja znački (engl. *token*) u UML dijagramu aktivnosti (engl. *activity diagram*).

- (1 bod) Opišite uvjete kretanja znački (engl. *token*) u UML dijagramu aktivnosti (engl. *activity diagram*).
- Rj. dio semantike bez grafičkog prikaza. Značka može predstavljati: upravljački tijek; objekt; podatak. Kreću se od izvorišta prema odredištu vezama ovisno o: ispunjenim uvjetima izvornog čvora, postavljenim uvjetima veza (engl. Edge guard conditions), preduvjetima ciljnog čvora.

- 11. (1 bod) Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?

- 11. (1 bod) Koji je cilj provođenja aktivnosti ispitivanja u procesu oblikovanja programske potpore?
- Rješenje: Otkrivanje informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema ispitivane programske podrške.
- Priznaje se i kraće rješenje: pronalaženje pogrešaka.

- 14. (1 bod) Navedite osnovno svojstvo razina u općenitoj višerazinskoj arhitekturi (engl. *n-tier architecture*).

- 14. (1 bod) Navedite osnovno svojstvo razina u općenitoj višerazinskoj arhitekturi (engl. *n-tier architecture*).
- Rj. Predstavlja proširenje raspodijeljene arhitekture klijent – poslužitelj. Razina zatvara (skriva, enkapsulira) skup usluga i implementacijske detalje niže razine o kojoj ovisi.

- 15. (1 bod) Što određuje tijek izvođenja programa u arhitekturi protoka podataka (engl. *data flow*)?



- 15. (1 bod) Što određuje tijek izvođenja programa u arhitekturi protoka podataka (engl. *data flow*)?
- Rj. Redoslijed izvršavanja instrukcija nije određen programskim brojiлом već već je nedeterminističan, odnosno ovisan o podacima koji su raspoloživi aktorima (engl. functional unit - FUs). Pomakom podataka aktivira se daljnje upravljanje.

- 16. (1 bod) U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. *version control, revision control, source control*). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?

- 16. (1 bod) U timskom razvoju programske potpore pojedine dijelove istodobno razvija više članova tima. Tim koristi GITLab sustav za upravljanje inačicama datoteka (engl. *version control, revision control, source control*). Gdje su smješteni podaci izvornog koda koje mijenjaju članovi tima tijekom programiranja (obrazložite odgovor)? Da li je moguć istodoban rad više članova tima na izvornom kodu?
- Rj. u lokalnim repozitorijima kod članova tima, moguć je istodoban rad.

- 17. (1 bod) Definirajte što znači da je skup logičkih formula  $\Gamma$  konzistentan.

- Oblikovanje programske podpore Stranica 6/9

- 18. (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:
- "Na nekom fakultetu postoji samo jedan izborni predmet 'Ekspertni sustavi'."

Rj.

$F(x)$  = x je fakultet

$IP(x,y)$  = x je izborni predmet na y

$=(x,y)$  = x je jednako y

'Ekspertni sustavi' = konst.

$\exists x (F(x) \wedge IP('Ekspertni sustavi', x) \wedge \neg \exists y (IP(y,x) \wedge \neg (=('Ekspertni sustavi',y))))$

alternativno:

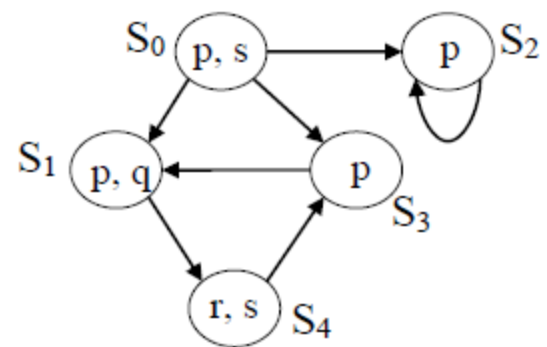
$\exists x (F(x) \wedge IP('Ekspertni sustavi',x) \wedge \forall y (IP(y,x) \Rightarrow (=('Ekspertni sustavi',y))))$

- 19. (1 bod) Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic):
- "Uvijek nakon req=1 konačno dođe ack=1."



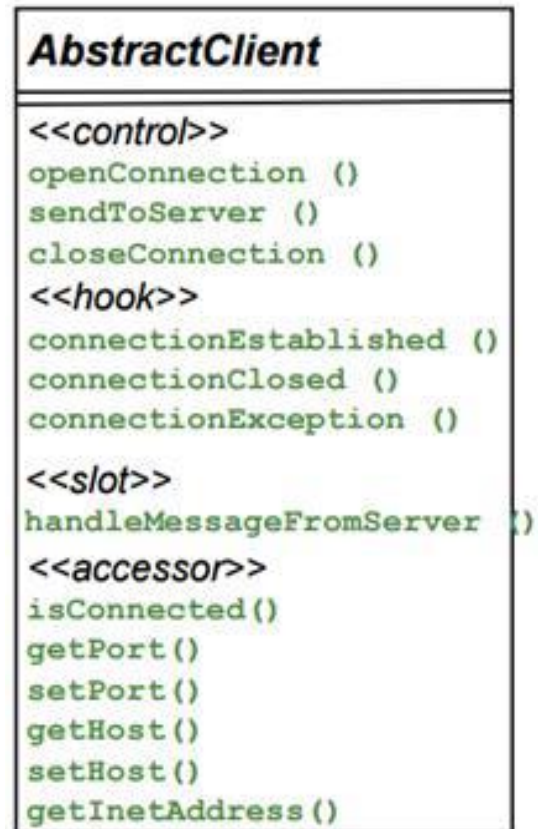
- 19. (1 bod) Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. Computational Tree Logic):
- "Uvijek nakon req=1 konačno dođe ack=1."
- Rj.  $AG (req=1 \Rightarrow AF ack=1)$ .

20. (1 bod) Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu  $\mathbf{A} (p \mathbf{U} r)$ .



- Rj. Takva stanja su =  $S_1, S_3, S_4$

(1 bod) UML dijagramom razreda prikazan je apstraktni razred klijenta s navedenim stereotipima metoda u radnom okviru klijentsko-poslužiteljske arhitekture. Koje metode programer može ali i ne mora implementirati? Čemu služi ta vrsta metoda?



- Rj. `connectionEstablished()`, `connectionClosed()`, `connectionException()`.
- To su metode koje se mogu po potrebi redefinirati. U radnom okviru za njih postoji minimalna implementacija.

- (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*):
  - a) (1 bod) Navedite korake ispitivanja.
  - b) (1 bod) Na primjeru jednog ulaznog 4-znamenkastog broja u intervalu [7000, 9997], navedite potrebne minimalne ispitne slučajeve.

1. Odredi particije za sve ulazne varijable.
2. Za sve particije odaberi vrijednosti ispitivanja.
3. Definiraj ispitne slučajeve koristeći odabrane vrijednosti.
4. Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.

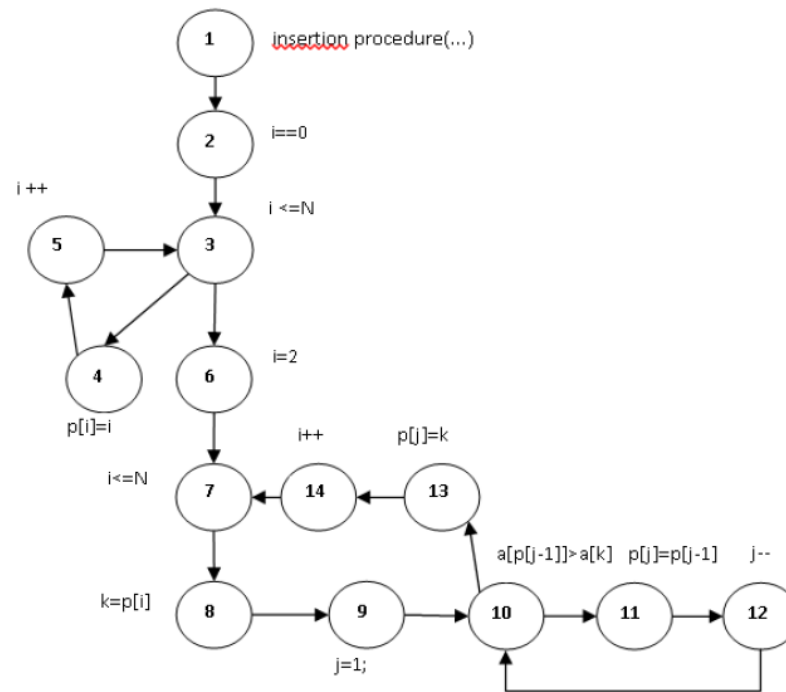
Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	6999	ne prolazi
IS2	7000	prolazi
IS3	8001	prolazi
IS4	9997	prolazi
IS5	9998	ne prolazi

13. (2 boda) Za sljedeću funkciju:

- a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.
- b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

```
01. insertion_procedure (int a[], int p [], int N)
02. {
03.     int i,j,k;
04.     for (i=0;i<=N; i++)
05.         p[i] = i;
06.     for (i=2;i<=N; i++)
07.     {
08.         k=p[i];j=1;
09.         while (a[p[j-1]] > a[k]) {
10.             p[j] = p[j-1];
11.             j--
12.         }
13.         p[j] = k;
14.     }
15. }
```





$$CV(G) = \text{Lukovi} - \text{Čvorovi} + 2 \cdot P = 16 - 14 + 2 = 4$$

Teorija: s ispita, ovdje zanimljivi zadaci  
ZI 18/19 A

13. (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*): a) (1 bod) Navedite korake ispitivanja.
- b) (1 bod) Na primjeru jednog ulaznog 3-znamenkastog broja u intervalu  $[700, 997]$ , navedite potrebne minimalne ispitne slučajeve.

13. (2 boda) Za funkcijsko ispitivanje ekvivalentnim particijama (engl. *equivalence partition*): a) (1 bod) Navedite korake ispitivanja.  
b) (1 bod) Na primjeru jednog ulaznog 3-znamenkastog broja u intervalu [700, 997], navedite potrebne minimalne ispitne slučajeve.

Rj.

1. Odredi particije za sve ulazne varijable.
2. Za sve particije odaberi vrijednosti ispitivanja.
3. Definiraj ispitne slučajeve koristeći odabrane vrijednosti.
4. Odredi očekivane izlaze za odabrane ispitne slučajeve i provedi ispitivanje.

Ispitni slučaj	ulazna vrijednost	očekivani izlaz
IS1	699	ne prolazi
IS2	700	prolazi
IS3	801	prolazi
IS4	997	prolazi
IS5	998	ne prolazi

\*priznaju se i dodatni ispitni slučajevi: (npr. 500, ne prolazi) (npr. 1200, ne prolazi)

14. (2 boda) Za sljedeću funkciju:

- a) (1 bod) Nacrtajte graf tijeka programa (engl. *control flow graph*). Uz program navedite odgovarajuće oznake na grafu.
- b) (1 bod) Odredite gornju granicu broja ispitnih slučajeva koja jamči potpuno pokrivanje svih naredbi programa. Navedite formulu i izračunajte.

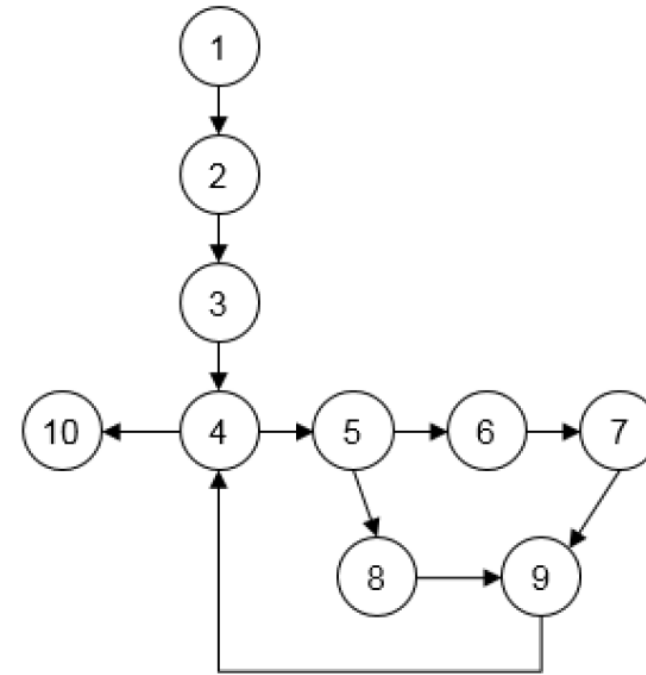
```
public void TDM(A[], x, y , z , B[])
{
    int i = x ;
    int j = y ;

    for (int k = x ; k < z ; k++) {
        if (i < y && (j >= z || A[i] <= A[j])) {
            B[k] = 7*A[i];
            i = i++;
        } else {
            B[k] = 4+A[j++] / 2;
        }
    }
    return;
}
```

Rj.

```
public void TDM(A[], x , y , z , B[])
{
    int i = x ; (1)
    int j = y ; (2)

    for (int k = x (3); k < z (4); k++ (9)) {
        if (i < y && (j >= z || A[i] <= A[j])) (5) {
            B[k] = 7*A[i]; (6)
            i = i++; (7)
        } else {
            B[k] = 4+A[j]/2; (8)
        }
    }
    return; (10)
}
```



b)  $CV(G) = 11 - 10 + 2 \cdot 1 = 3$

---

- (1 bod) Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:
- *"U poslužitelju Sparc T8-4 postoje barem dva procesora."*

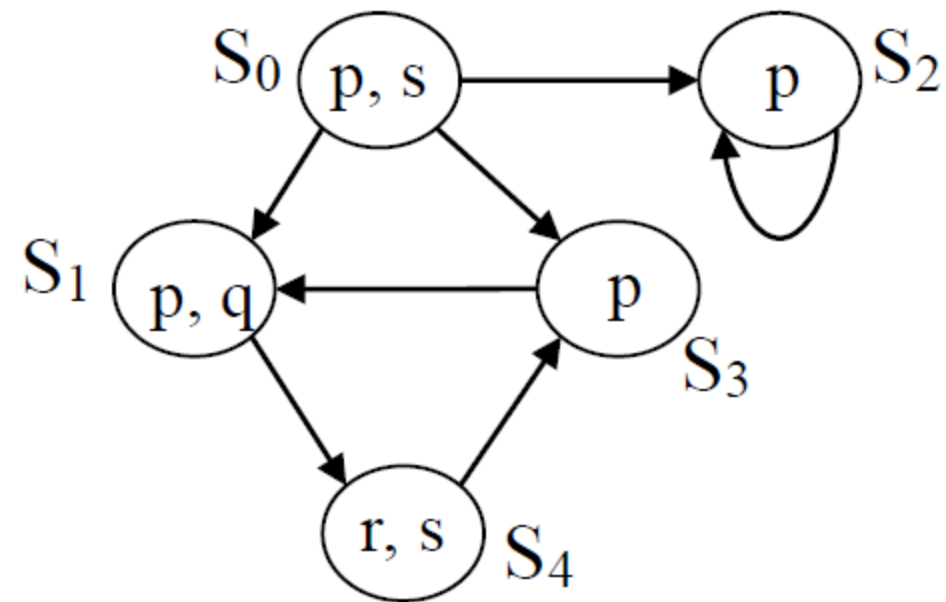
- $R_j$ .  $\text{poslužitelj}(x)$  -  $x$  je poslužitelj
- $\text{procesor}(x)$  -  $x$  je procesor
- $\text{postoji\_u}(x,y)$  -  $x$  postoji u  $y$
- $=(x,y)$  -  $x$  je jednako  $y$
- Sparc T8-4 - konstanta
- $\text{poslužitelj}(\text{Sparc T8-4}) \Rightarrow \exists x \exists y (\text{procesor}(x) \wedge \text{procesor}(y) \wedge \text{postoji\_u}(x, \text{Sparc T8-4}) \wedge \text{postoji\_u}(y, \text{Sparc T8-4}) \wedge \neg =(x,y))$



- (1 bod) Prevedite sljedeću rečenicu prirodnog jezika u formalizam logike CTL (engl. *Computational Tree Logic*):
- "Iz početnog stanja postoji put na kojem konačno vrijedi  $\text{stop}=1$ , nakon čega dalje nikad ne vrijedi  $\text{start}=1$ ."

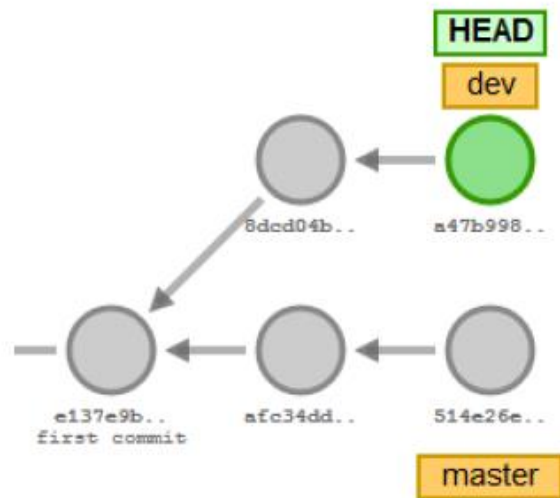
- Rj.  $EF (stop = 1 \wedge AG \neg(start=1))$  priznaje se i:  $EF (stop = 1 \wedge AX AG \neg(start=1))$

- (1 bod) Za zadani model implementacije Kripke strukturom  $M$  prema slici potrebno je odrediti skup svih stanja koja zadovoljavaju formulu **EG (p)**.

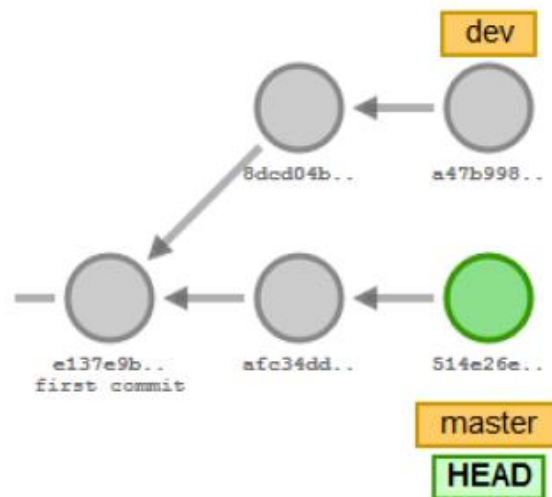


- Rj. Takva stanja su =  $S_0, S_2$

20. (1 bod) Kojom git naredbom repozitorij prelazi iz stanja sa slike A) u stanje sa slike B)?



A)



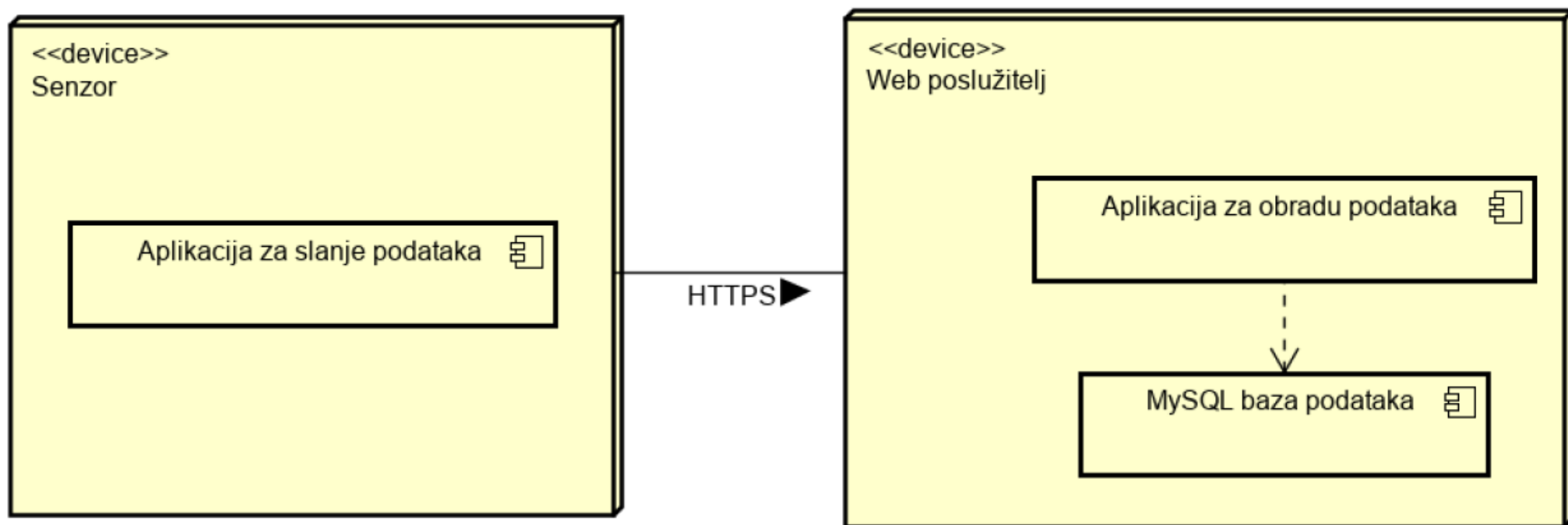
B)

# ZI 2020 A, OPET TEORIJA S ISPITA

- Git checkout master

- 8. (2 boda) Nacrtajte specifikacijski dijagram razmještaja koji prikazuje komunikaciju senzora s web poslužiteljem. Aplikacija za slanje podataka na senzoru spaja se HTTPS protokolom na aplikaciju za obradu podataka koja se nalazi na poslužitelju. Aplikacija za obradu podataka ovisi o MySQL bazi podataka koja se nalazi na istom poslužitelju.

Rj.



Napomena: komponente mogu imati stereotip, kao što je <<artefact>>



- (2 boda) Kod radnog okvira OCSF: a) (1 bod) Koliko najmanje dretvi treba biti prisutno na poslužiteljskoj strani, ako su na poslužitelj spojena tri klijenta? Napomena: zanemarite dretvu za komunikaciju s administratorom poslužitelja kojim se kontrolira rad poslužitelja.
- b) (1 bod) Koja je razlika između upravljačkih (<<control>>) metoda i metoda kopči (<<hook>>) kod razreda AbstractClient?

- a) 4 dretve – jedna koja osluškuje zahtjeve s klijenta i 3 za komunikaciju s klijentima.
- b) kontrolne metode imaju punu i dobro ispitanu implementaciju i ne mogu se nadjačati, dok metode kopči imaju trivijalnu implementaciju i služe da se redefiniraju (nadjačaju) u razredu koji nasljeđuje razred AbstractClient.

- Funkciju koja kao ulaze prima dva cijela broja u rasponu  $[-51, 100]$  potrebno je ispitati primjenom tehnike kombinacijskog ispitivanja (engl. *combination testing*). Izračunajte broj potrebnih ispitnih slučajeva.

- lazne vrijednosti -51..100:  $152 * 152 = 23\ 104$  ispitnih slučajeva.

- Na koji se način provjerava ispravnost dobivenog i očekivanog izlaza u xUnit (JUnit, NUnit...) radnim okvirima za ispitivanje programske potpore?

- Assert metodama

16. (2 boda) Za sljedeću funkciju:

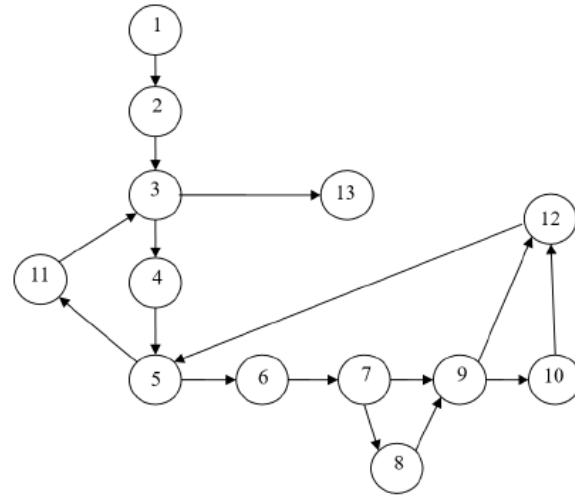
- a) (1 bod) Nacrtajte graf tijeka programa.
- b) (1 bod) Odredite njezinu ciklomatsku složenost. Navedite formulu za njezin izračun.

```
public Flux computeFlux(double[] q, double alpha, int sizex, int sizey) {  
    Flux flux = new Flux(sizex, sizey);  
    int index;  
    for (int i = 0; i < sizex; i++) {  
        for (int j = 0; j < sizey; j++) {  
            index = getIndex(i,j);  
            if (j != 0) {  
                flux.addx(index, alpha*(q[index]-q[index-1])/2);  
            }  
            if (i != 0) {  
                flux.addy(index, alpha*(q[index]-q[index-sizex])/2);  
            }  
        }  
    }  
    return flux;  
}
```

Rj.

a)

```
Flux flux = new Flux(sizeX, sizeY); (1)
int index;
for (int i = 0; (2) i < sizeX; (3) i++ (11)) {
    for (int j = 0; (4) j < sizeY; (5) j++ (12)) {
        index = getIndex(i,j); (6)
        if (j != 0) { (7)
            flux.addx(index, alpha*(q[index]-q[index-1])/2); (8)
        }
        if (i != 0) { (9)
            flux.addy(index, alpha*(q[index]-q[index-sizeX])/2); (10)
        }
    }
}
return flux; (13)
```





- (1 bod) Prevedite rečenicu prirodnog jezika u formulu vremenske logike CTL:
- *„Na svakom putu od početnog stanja konačno dolazimo u stanje od kojeg nadalje stalno vrijedi p.“*

- Rj.
- a) AF AG p
- Napomena: priznaje se i rješenje AF AX AG p

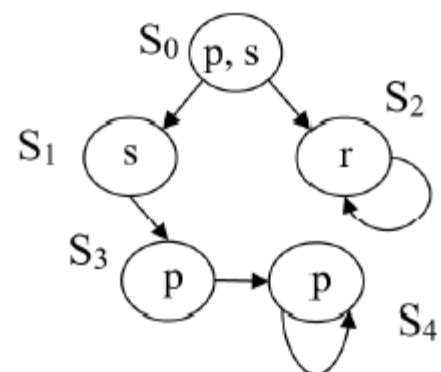
- Definirajte potrebne predikate i konstante te preslikajte rečenicu u dobro definiranu formulu predikatne logike prvoga reda:
- *"Svaki klijent spojen je s barem jednim poslužiteljem."*

- $R_j$ .
- $K(x)$  =  $x$  je klient
- $P(x)$  =  $x$  je poslužitelj
- $S(x,y)$  =  $x$  je spojen s  $y$
- $\forall x (K(x) \Rightarrow \exists y (P(y) \wedge S(x,y)))$

- 18. (1 bod) Što mora biti zadovoljeno za interpretacije skupa formula **G**, tako da možemo tvrditi da vrijedi **G** |= **P** tj. da je formula **P** logička posljedica skupa **G**.

- Rj. Svaka interpretacija koja za skup  $G$  daje istinitost mora i za  $P$  dati istinitost.

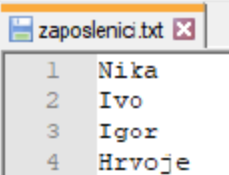
20. (1 bod) Za zadani model implementacije prikazan Kripke strukturom prema slici potrebno je navesti skup stanja koja zadovoljavaju formulu **EG p**.



- S3,s4

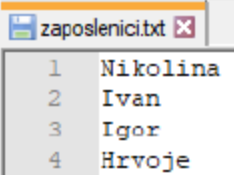


(1 bod) Trenutno se nalazimo na grani master. Osim nje, postoji i grana develop. Obje grane sadrže istu datoteku „zaposlenici.txt“ različitog sadržaja (prikazano na donjoj slici). Napišite kako će izgledati datoteka na trenutnoj grani nakon izvođenja naredbe „git merge develop“.



```
1 Nika
2 Ivo
3 Igor
4 Hrvoje
```

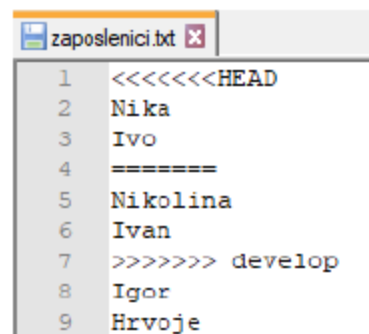
grana master



```
1 Nikolina
2 Ivan
3 Igor
4 Hrvoje
```

grana develop

Rj.



```
1 <<<<<<<HEAD
2 Nika
3 Ivo
4 =====
5 Nikolina
6 Ivan
7 >>>>>>> develop
8 Igor
9 Hrvoje
```

Napomena: ne priznaju se rješenja u kojima se navodi izgled datoteke **nakon** donošenja odluke prilikom konflikta

by mačka