Dežurni nastavnik: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Testiranje softvera (SI3TS)
Nastavnik: doc. dr Dragan Bojić
Asistent: dipl. ing. Dražen Drašković
Ispitni rok: Oktobar 2012.
Datum: 23.09.2012.

Kandidat:* _____

Broj indeksa:* _____

Ispit traje 3 sata, prvih sat vremena nije dozvoljeno napuštanje ispita.
Upotreba literature nije dozvoljena.

Zadatak 1	_____/6	*Zadatak 5*	_____/8
Zadatak 2	_____/6	*Zadatak 6*	_____/8
Zadatak 3	_____/16	*Zadatak 7*	_____/8
Zadatak 4	_____/8		

Ukupno na ispitu: **_____/60** ***Ukupno na domaćem*:*** **_____/40**

Rok u kome je odbranjen domaći*: **_____** (primer: januar 2012)

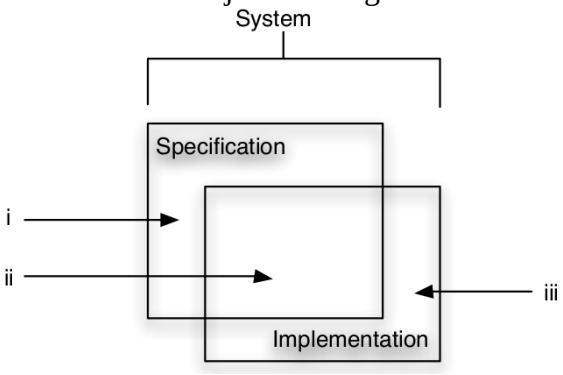
Ukupno: _____/100

Ocena: _____ (_____)

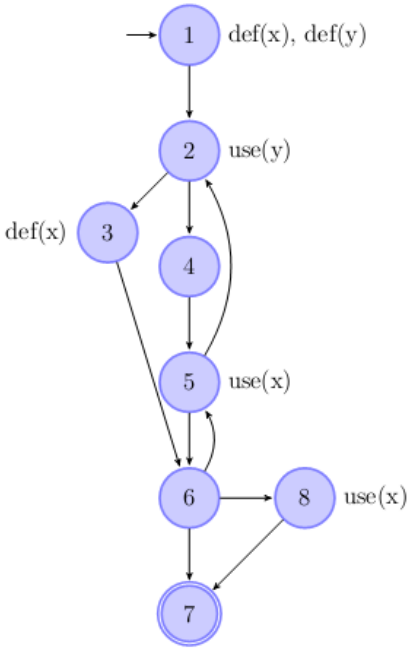
Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**. * popunjava student.

---

1. [6] Jedan sistem je implementiran na osnovu specifikacije, ali se dogodilo da se implementirane funkcije ne poklapaju sasvim sa specificiranim, kao što je na slici prikazano skupovima funkcija.
   a) Navesti konkretne specifikacije i implementacije kada se pojavljuju situacije I i III.
   b) Kojom vrstom testiranja bi se mogla otkriti situacija I, a kojom situacija III?



2. [6] Da li dati skup putanja [1-2-3-6-8-7], [1-2-4-5-6-7], [1-2-4-5-2-3-6-5-6-7] pokriva u datom grafu (uz odgovore NE dati obrazloženje):
   a) sve putanje
   b) sve bazične putanje
   c) sve grane
   d) sve čvorove
   e) sve definicije (all defs)
   f) sve upotrebe (all-uses)
   g) sve du putanje (all-du paths)



3. [16] a) Neka je dat deo softverskog sistema koji služi za unos artikala u prodavnici obuće i spisak različitih veličina određene obuće. Specifikacija navodi da artikli treba da budu alfabetski znaci od 2 do 15 karaktera. Svaka veličina može biti u opsegu od 26 do 55, i predstavlja se samo celim brojevima. Veličine se unose u rastućem poretku (prvo manje veličine). Maksimalno pet veličina se može uneti za svaki artikal. Format unosa je: naziv artikla, sledi zarez, a zatim sledi spisak veličina. Zarez se takođe koristi da razdvoji veličine. Blanko znaci treba da se zanemare na ulazu. Napisati sve klase ekvivalencije za ovaj deo softverskog sistema. Napisati skup test primera koji će pokriti ove klase ekvivalencije (za svaki primer navesti ulazne podatke i očekivani izlaz).

b) Metodom graničnih slučajeva testirati formu fakulteskog sistema ETF za prikazivanje rasporeda časova i dati minimalan skup test primera, ako se zna da se evidencija studenata vodi od 1986.godine, da studenata osnovnih studija ima maksimalno 560 i dodeljuju im se brojevi indeksa od 1. Master studenti i studenti doktorskih studija se upisuju od 2007.godine, pri čemu u svakoj generaciji prvi master student ima indeks 3001, a prvi student doktorskih studija ima indeks 5001.

4. [8] Naznačiti koliko ukupno definicija, c-upotreba i p-upotreba po svakoj od navedenih celobrojnih promenljivih - *domaci*, *poeni*, *ukupno*, *brojac_studenata*, ima u sledećem Java programu, koji računa ocene na predmetu.
Napomena: pretpostaviti da su sve metode *citajDomaci*, *citajPoeneIspit* i *citajPoeneProjekat* definisane.

```
package ispit;

public class Ocena {
        public static void main(String args[]) {
(1)           int ukupno_studenata = args.length;
(2)           int brojac_studenata = 0;
(3)           while (brojac_studenata < ukupno_studenata) {
(4)                 System.out.println(args[brojac_studenata]);
(5)                 int broj_domacih = 4;
(6)                 int domaci = 0, ukupno = 0;

(7)                 while (domaci < broj_domacih) {
(8)                    int poeni = citajDomaci(args[brojac_studenata],
                       domaci);
(9)                    if (poeni < 0)
(10)                         System.out.println("Nekorektan unos");
(11)                   else {
(12)                         ukupno += poeni;
(13)                         domaci++;
(14)                   }
(15)                }
(16)                int poeni = citajPoeneIspit(args[brojac_studenata]);
(17)                if (poeni < 0)
(18)                      System.out.println("Nekorektan unos");
(19)                else if (poeni >= 31 && poeni <= 50)
(20)                      ukupno += poeni;

(21)                poeni = citajPoeneProjekat(args[brojac_studenata]);
(22)                if (poeni >= 10 && poeni <= 30)
(23)                      ukupno += poeni;

(24)                System.out.println("Ukupno poena " + ukupno);

(25)                if (ukupno >= 91)
(26)                      System.out.println("Ocena 10");
(27)                else if(ukupno >= 81)
(28)                      System.out.println("Ocena 9");
(29)                else if(ukupno >= 71)
(30)                      System.out.println("Ocena 8");
(31)                brojac_studenata++;
(32)          }
(33)    }
(34)}
```

5. [8] Dat je algoritam koji sortira niz od n elemenata. Testirati sve petlje u ovoj proceduri i detaljno obrazložiti:

```
public class HeapSort{
public static void main(String arg[]){
        int a[]={3,30,4,12,1,2,34,22,4,3,5,8};
        System.out.println("\n\nUlazni niz:");
        for(int i=0;i<a.length;i++){
                System.out.print(" "+a[i]);
        }
        int N = a.length;
        for (int k = N/2; k > 0; k--) {
            downheap(a, k, N);
        }
        do {
             int T = a[0];
             a[0] = a[N - 1];
             a[N - 1] = T;
            N = N - 1;
           downheap(a, 1, N);
        } while (N > 1);
         System.out.println("\n\n\nSortirani niz:");
        for(int i=0;i<a.length;i++){
            System.out.print(" "+a[i]);
        }
}
 static void downheap(int a[], int k, int N) {
        int T = a[k - 1];
        while (k <= N/2) {
             int j = k + k;
             if ((j < N) && (a[j - 1] < a[j])) {
                 j++;
             }
             if (T >= a[j - 1]) {
                 break;
             } else {
                  a[k - 1] = a[j - 1];
                  k = j;
             }
        }
        a[k - 1] = T;
    }
}
```

a) Testirati sve petlje u ovoj proceduri i detaljno obrazložiti.
Da li je ispravno napisana ova procedura?

b) U kom slučaju ova procedura daje najbolje performanse?

6. [8] Neka je dat sledeći deo programskog koda.
   a) Odrediti sve LCSAJ za datu metodu.
   b) Odrediti minimalan skup testova koji pokrivaju sve LCSAJ.

```
   public void uredi() throws Greska{
1.      int n = niz.length;
2.      for (int i=1; i<n; i++) {
3.          double p = niz[i];
4.          int j = i - 1;
5.          while (j>=0 && p < niz[j]) {
6.                niz[j+1] = niz[j--];
7.                prikazi(); //funkcija za prikazivanje niza
8.          }
9.      niz[j+1] = p;
10.     prikazi();
11.     }
12.   }
```

Napomena: pretpostaviti da je kod u metodi `prikazi()` linearna sekvenca i da ne postoje skokovi.

7. [8] Neka je data sledeća funkcija:

```
public int funkcija(int x, int y) {
      int rezultat = 1;
      int brojac = y;
      while(brojac > 0){
            rezultat = rezultat * x;
            brojac--;
      }
      return(rezultat);
}
```

a) Koliko će mutanata generisati ova funkcija kada se primeni ABS (absolute value) operator? Napisati sve mutante koji su generisani ABS operatorom, napisati koji su live/killed i izračunati mutacioni rezultat.

b) Razmotriti sledeći mutant, koji se dobija promenom linije 5:

```
rezultat = rezultat * abs(x);
```

Napraviti test primer koji razlikuje rezultat od ovog mutanta ili pokazuje da je ekvivalentan ovoj funkciji.

c) Napraviti najmanje jedan mutant koji je ekvivalentan funkciji.