Dežurni nastavnik: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Testiranje softvera (13S113TS)
Nastavnici: Prof. dr Dragan Bojić, doc. dr Dražen Drašković
Ispitni rok: Drugi kolokvijum (decembar 2022.)
Datum: 6.12.2022.

Kandidat:* _____

Broj indeksa:* _____ *E-pošta*:* _____*@student.etf.bg.ac.rs*

„Danas polažete dva ispita: iz struke i iz poštenja. Ako treba jedan da padnete, neka to bude iz struke, jer ćete kao pošteni ljudi, lako savladati struku, a ako padnete na poštenju, nećete ga savladati nikad.“

Kolokvijum traje 120 minuta, a u prvih 60 minuta nije dozvoljeno napuštanje kolokvijuma. Upotreba literature nije dozvoljena.

Zadatak 1 _____/5
Zadatak 2 _____/4
Zadatak 3 _____/5
Zadatak 4 _____/6

Ukupno na kolokvijumu: _____/20

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 i precizno**.

* popunjava student.

Programski kod uz zadatke 1 i 2:

```
(01) float calculateInterestRate(Account account) {
(02)
(03)  final float BUSINESS_RATE     = 3.0f;
(04)  final float INDIVIDUAL_RATE   = 1.0f;
(05)  final float BONUS             = 0.5f;
(06)  final float MAX_RATE          = 7.0f;
(07)
(08)  float rate = 0.0f;
(09)
(10)  switch (account.Type()) {
(11)        case BUSINESS:
(12)              rate = BUSINESS_RATE;
(13)              break;
(14)
(15)        case INDIVIDUAL:
(16)              rate = INDIVIDUAL_RATE;
(17)              if (account.getCustomer().isSeniorCitizen()) {
(18)                    rate += BONUS;
(19)              }
(20)              break;
(21)
(22)        default:
(23)              // should be impossible
(24)              assert(false);
(25)              break;
(26)  }
(27)
(28)  if (account.getYearsOpen() > 5) {
(29)        rate += BONUS;
(30)  }
(31)
(32)  if (account.getBalance() > 100000.0f) {
(33)        rate += ((account.getBalance() / 50000.0f) * BONUS);
(34)  }
(35)
(36)  if (rate > MAX_RATE) {
(37)        rate = MAX_RATE;
(38)  }
(39)
(40)  return rate;
(41) }
```

1. [5] [1] a) Odrediti broj ciklomatske kompleksnosti za dati metod *calculateInterestRate()*.

   [2] b) Odrediti bazični skup putanja (navesti putanje kao sekvence brojeva linija).

   [2] c) Odrediti konkretne test primere za svaku putanju iz tačke b).

2. [4] [2] a) Odrediti sve DU lance za promenljivu rate u metodu *calculateInterestRate()*.

   [2] b) Navesti koji DU lanci iz tačke a) NISU pokriveni testovima iz tačke c) zadatka 1.

3. [5] Direktor Cvetković je kupio polovni automobil *Toyota* za svoje preduzeće i pokušava da testira bord kompjuter na osnovu uputstva. Kompjuterom se upravlja preko ručice koja ima jedno dugme X i pokretni deo ručice, koji može da ide unapred i unazad (u daljem tekstu dugmad: + i -). Bord kompjuter prikazuje početno stanje trenutne kilometraže automobila (u jedinici: km). Kratkim pritiskom na X prelazi se u stanje tempomata, a dugačkim pritiskom na X prelazi se u stanje predikcije moguće kilometraže, koju bi automobil mogao da pređe (pri trenutnom stanju goriva u rezervoaru i prosečnoj potrošnji). Stanje tempomata prikazuje trenutnu Tempo_brzinu iz memorije automobila. Okretanjem ručice unapred (dugme +) *Tempo_brzina* se povećava za 10 km/h, a unazad okretanjem (dugme –) *Tempo_brzina* se smanjuje za 5 km/h. Iz tog stanja tempomata, kratkim pritiskom na X se prelazi u stanje koje prikazuje prosečnu potrošnju automobila u gradu, dok će se dugačkim pritiskom na X preći u stanje koje prikazuje prosečnu potrošnju automobila van grada. Iz oba stanja prosečne potrošnje, kratkim pritiskom na X se prelazi u novo stanje kompjutera za prikaz prosečne ukupne potrošnje goriva. Sve potrošnje goriva su izražene u jedinici L/100km. Pri prikazu prosečne ukupne potrošnje, dugačkim pritiskom na dugme X, ponovo se prelazi u već opisano stanje predikcije moguće kilometraže koje auto može da pređe, a duplim kratkim pritiskom na X, kompjuter se vraća u početno stane. Iz stanja predikcije moguće kilometraže, jednim dugačkim pritiskom dugmeta X, kompjuter se vraća takođe u početno stanje, a duplim kratkim pritiskom na X, prikazuje se prosečna ukupna potrošnja auta. Svi prikazi potrošnje auta, kao i trenutna vrednost tempomata, može da se poništi na 0 duplim kratkim pritiskom na dugme X. Cvetković želi da testira tempomat obavezno pri brzinama vožnje na autoputu (130 km/h), međugradske (70 km/h) i gradske vožnje (45 km/h), kao i sa polovičnim i punim rezervoarom, da bi video da li korektno radi predikcija moguće kilometraže.

a) [3] Nacrtati konačan automat koji odgovara ovoj specifikaciji.

Odrediti sve 0-izmene i 1-izmene za dati bord kompjuter automobila.

b) [2] Odrediti skup test primera koji pokrivaju 1-izmene određene u tački a). Za svaki test primer navesti koje 1-izmene pokriva.

Nije potrebno navoditi nelegalne tranzicije.

Rešenje a) - graf:

Rešenje a) – tabela 0-izmena i 1-izmena:

Rešenje b) – test primeri:

4. [6] Data je metoda za proračun honorara u preduzeću, realizovana u programskom jeziku *Java* (pozivom *new Date()* formira se instanca sa trenutnim datumom):

```
   public static double izracunaj(int godine, int brojClanovaPorodice,
                double osnovica, double minIznos, boolean povlascen) {
1      double stopa;
2      Date datum = new Date();
3      Calendar cal = Calendar.getInstance();
4      cal.setTime(datum);
5      int preostalo = 12 - cal.get(Calendar.MONTH);
6
7      if(!povlascen) {
8          stopa = osnovica;
9      } else {
10         if(povlascen && godine >= 35) {
11             stopa = osnovica * 0.80;
12         } else {
13             stopa = osnovica * 0.65;
14         }
15     }
16     osnovica = stopa * preostalo;
17     while(brojClanovaPorodice >= 2 && osnovica > minIznos) {
18         if(godine >= 21) {
19             osnovica = osnovica - 10;
20         } else {
21             osnovica = osnovica - 5;
22         }
23         brojClanovaPorodice = brojClanovaPorodice - 1;
24     }
25     return osnovica;
26 }
```

a) [3] Za datu metodu odrediti <u>minimalni</u> skup test primera, koji pokriva sve odluke i uslove.

Broj_TP[1]	*datum izvršenja*	*godine*	*brojClan.*	*osnovica*	*minIznos*	*povlascen*	**Pokr.odluke/uslovi**	**Izlaz**

b) [3] Napisati sve linearne sekvence korišćenjem tehnike LCSAJ (*Linear Code Sequence and Jump*) i nakon toga napisati da li test primeri realizovani pod tačkom a) obuhvataju sve takve sekvence, i koliki je procenat pokrivenosti realizovanih sekvenci?

Da li je moguće realizovati još neki test primer koji bi pokrio preostale sekvence?

Sekvence pisati u obliku uređene trojke (*START_SEQ*, *END_SEQ*, *JUMP*).
Upariti sve pronađene sekvence sa test primerima koje ćete realizovati.

LSCAJ sekvence:

R.br.[1]	Start	End	Jump
L1			
L2			
L3			
L4			
L5			
L6			
L7			
L8			
L9			
L10			
L11			
L12			
...			

Test primeri:

[1] *Broj redova u tabelama ne mora da odgovara tačnom broju test primera ili tačnom broju LCSAJ sekvenci.*