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: Drugi kolokvijum (novembar 2013.)
Datum: 26.11.2013.

*Kandidat**:* _____

*Broj Indeksa**:* _____ *E-mail*:* _____

Kolokvijum traje 1.5 sat, prvih sat vremena nije dozvoljeno napuštanje kolokvijuma.
Upotreba literature nije dozvoljena.

Zadatak 1 _____/5
Zadatak 2 _____/10
Zadatak 3 _____/5

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.

1. [5] Potrebno je integraciono testirati Java program *Palindrome*, koji proverava da li je zadati string palindrom, tj. da li se čita isto unazad, kao i unapred. Ulazi za testiranje su "ana" i "baraba" i program se pokreće ručno (nije potreban drajver). Nije na raspolaganju implementacija funkcije *IsPalindrome*.

   (a) Napisati dummy funkcije *IsPalindrome*.
   (b) Napisati stub funkcije *IsPalindrome*.

```
import java.util.*;

class Palindrome {

    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter a string to check if
                            it is a palindrome");
        String original = in.nextLine();
        if (IsPalindrome(original)) {
            System.out.println("Entered string is a palindrome.");
        } else {
            System.out.println("Entered string is not a palindrome.");
        }
    }
}
```

2. [10] Neka je dat Java program koji radi sa matricama. Matrica *Matrix* je dvodimenzionalni niz celobrojnih podataka. Ona se definiše sa tri parametra: broj redova (*nrows*), broj kolona (*ncols*) i elementima niza (*data*):

```
public class Matrix {
    private int nrows;
    private int ncols;
    private double[][] data;

    public Matrix(double[][] dat) {
        this.data = dat;
        this.nrows = dat.length;
        this.ncols = dat[0].length;
    }
    public Matrix(int nrow, int ncol) {
        this.nrows = nrow;
        this.ncols = ncol;
        data = new double[nrow][ncol];
    }
    public boolean isSquare(){
        return this.nrows==this.ncols;
    }
    public int size(){
        return this.nrows;
    }
    public double getValueAt(int i, int j){
        return this.data[i][j];
    }
    public void setValueAt(int r, int c, double d){
        this.data[r][c] = d;
    }
    public int getNrows(){
        return this.nrows;
    }
    public int getNcols(){
        return this.ncols;
    }
    public int changeSign(int i){
        if(i%2==0) return 1;
        else return -1;
    }
}
```

a) Za sledeću metodu determinant(*Matrix matrix*) kojom izračunavamo determinantu matrice i koja ima 4 moguća izlaza iz metode, napisati test primere koje pokrivaju svaki iskaz.

```
public    static    double    determinant(Matrix    matrix)    throws
NoSquareException {
        if (!matrix.isSquare()) {
            throw new NoSquareException("GRESKA! ");
        }
        if (matrix.size() == 1) {
            return matrix.getValueAt(0, 0);
        }
        if (matrix.size() == 2) {
            return (matrix.getValueAt(0, 0)*matrix.getValueAt(1, 1))
               - (matrix.getValueAt(0, 1) * matrix.getValueAt(1, 0));
        }
        double sum = 0.0;
        for (int i = 0; i < matrix.getNcols(); i++) {
            sum += matrix.changeSign(i) * matrix.getValueAt(0, i) *
                   determinant(createSubMatrix(matrix, 0, i));
        }
        return sum;
    }
```

b) Za metodu *createSubMatrix*, koja formira podmatricu, odrediti sve LCSAJ, i minimalni skup test primera (odnosno matrica), koji pokriva sve LCSAJ za tu metodu.

```
public static Matrix createSubMatrix(Matrix matrix,
int excluding_row, int excluding_col) {
1)Matrix mat = new Matrix(matrix.getNrows()-1, matrix.getNcols()-1);
2)    int r = -1;
3)    for (int i=0; i<matrix.getNrows(); i++) {
4)        if (i==excluding_row)
5)            continue;
6)            r++;
7)            int c = -1;
8)        for (int j=0; j<matrix.getNcols(); j++) {
9)            if (j==excluding_col)
10)               continue;
11)           mat.setValueAt(r, ++c, matrix.getValueAt(i, j));
12)       }
13)   }
14)   return mat;
15)}
```

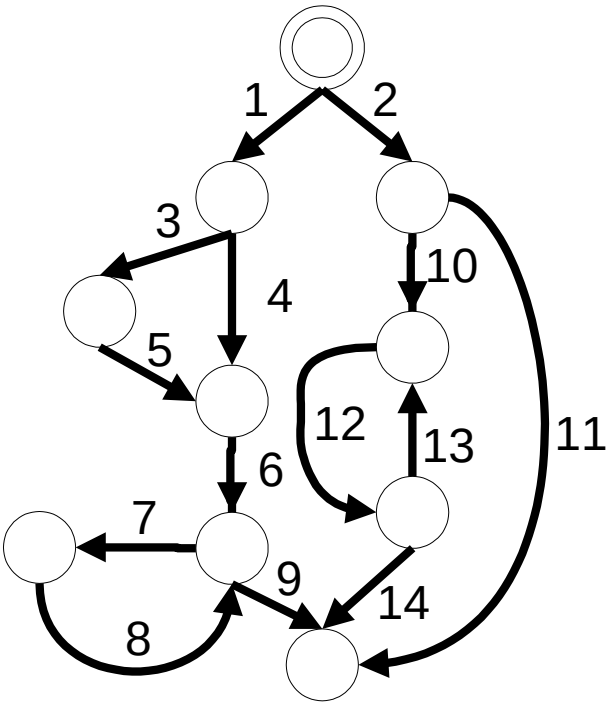
c) U metodi *createSubMatrix*, koja formira podmatricu neke matrice, za sve promenljive navedene u tabeli ispod, odrediti mesta definisanja i upotrebe tih promenljivih.

Pretpostaviti da su u glavnom programu, koji poziva ovu metodu, ispravno definisane (inicijalizovane) promenljive, koje se koriste kao ulazni argumenti metode, što je označeno naredbama iz glavnog programa (Gn), pa i njih navesti u tabeli, ukoliko u tim naredbama postoje definicije/upotrebe:

```
G1) Matrix m1 = new Matrix(3,4);
G2) double[][] dat = {{1,2,3},{4,5,6},{7,8,9}};
G3) m1 = new Matrix(dat);
G4) m2 = createSubMatrix(m1,2,2);
```

Promenljiva	DEF	C-USE	P-USE
mat			
matrix			
r			
c			
i			
j			

3. [5] Neka je dat graf kao na slici, koji predstavlja jedan program.



a) Ako je u korenom čvoru ovog programa dat sledeći uslovni iskaz:

```
IF(koef<10 AND cena>250)
THEN… ELSE …
```

Napisati minimalan skup test primera za pokrivanje sledećih vrsta testiranja belom kutijom:

Vrsta testiranja	Test primeri
uslova (*Condition Coverage*)	
odluka i uslova (*Decision/Condition Coverage*)	
višestrukih uslova (*Multiple Condition Coverage*)	

b) Napisati formulu po kojoj se računa i izračunati *McCabe*-ov broj cikolomatske kompleksnosti ovog grafa:

V = _____ = _____

Napomena: objasniti sve simbole koje navedete u formuli.