JEDANAESTI ČAS RAČUNSKIH VJEŽBI IZ PRINCIPA PROGRAMIRANJA

1. Kreirati klasu **VEKTOR** koja ima dva člana podatka, cio broj **N**, koji predstavlja dužinu vektora, i vektor **X** od maksimalno 100 realnih brojeva. Za klasu implementirati konstruktore, mutatore, inspektore, metode za sabiranje i oduzimanje vektora, kao i kvadriranje elemenata vektora. Ako u slučaju sabiranja i oduzimanja vektori nisu istih dužina, onda odgovarajući metod treba da vrati vektor u kome je N=-1, kao sugestija programu da ova operacija nije uspjela.

   Realizacija klase:

```
CLASS VEKTOR
N: INTEGER
X[100]: FLOAT
VEKTOR()
VEKTOR(A: FLOAT)
KOPIRAJ(P: VEKTOR)
POSTAVI(I: INTEGER, A: FLOAT)
CONST STAMPAJ()
CONST SABERI(P: VEKTOR): VEKTOR
CONST ODUZMI(P: VEKTOR): VEKTOR
CONST KVADRIRAJ(): VEKTOR
ENDCLASS
```

   Realizacija metoda:

```
VEKTOR() FROM VEKTOR
I: INTEGER
N=10
I=1
WHILE I≤N
X[I]=0
I=I+1
ENDWHILE
```

```
VEKTOR(A:FLOAT) FROM VEKTOR
I: INTEGER
N=10
I=1
WHILE I≤N
X[I]=A
I=I+1
ENDWHILE
```

```
KOPIRAJ(P: VEKTOR) FROM VEKTOR
I: INTEGER
N=P.N
I=1
WHILE I≤N
X[I]=P.X[I]
I=I+1
ENDWHILE
```

```
POSTAVI(I: INTEGER, A: FLOAT) FROM VEKTOR
X[I]=A
```

```
STAMPAJ() FROM VEKTOR
I: INTEGER
I=1
WHILE I≤N
OUTPUT X[I]
I=I+1
ENDWHILE
```

```
SABERI(P: VEKTOR): VEKTOR FROM VEKTOR
I: INTEGER
Q: VEKTOR
IF P.N≠N
Q.N=-1
RETURN Q
ENDIF
Q.N=N
I=1
WHILE I≤N
Q.X[I]=X[I]+P.X[I]
I=I+1
ENDWHILE
RETURN Q
```

```
KVADRIRAJ(): VEKTOR FROM VEKTOR
I: INTEGER
Q: VEKTOR
Q.N=N
I=1
WHILE I≤N
Q.X[I]=X[I]*X[I]
I=I+1
ENDWHILE
RETURN Q
```

2. Kreirati klasu STUDENT koja ima sljedeće podatke: **IME**, **PREZIME**, **GU** (Godina Upisa), **GS** (Godina Studija), **BPI** (Broj Položenih Ispita) i **PROSJEK**. Formirati konstruktore, inspektore i mutatore za ovu klasu. Formirati i metodu **PROVJERA** koja provjerava da li su podaci o studentu pravilno unešeni, tj. da ne postoje studenti upisani prije 1960 ni poslije 2006, da ne postoje studenti koji studiraju na godini studija manjoj od 1 niti većoj od 5, da broj položenih ispita ne može biti manji od 0 ni veći od 40, a prosjek mora biti u granicama od 6 do 10.

Realizacija klase:

```
CLASS STUDENT
IME[20], PREZIME[20]: CHAR
GU, GS, BPI: INTEGER
PROSJEK: FLOAT
STUDENT()
STUDENT(A[]:CHAR, B[]:CHAR, C: INTEGER, D: INTEGER, E: INTEGER, F:FLOAT)
CONST STAMP_STUD()
CONST DAJ_BPI(): INTEGER
CONST DAJ_PROS(): FLOAT
POV_GS()
AZURIRANJE(A: INTEGER)
PROVJERA(): INTEGER
ENDCLASS
```

Realizacija metoda:

```
STUDENT() FROM STUDENT
GU=0
GS=0
BPI=0
PROS=0
```

```
STUDENT(A[]:CHAR, B[]:CHAR, C: INTEGER, D:
INTEGER, E: INTEGER, F:FLOAT) FROM STUDENT
I: INTEGER
I=1
WHILE A[I]≠'\0'
IME[I]=A[I]
I=I+1
ENDWHILE
IME[I]='\0'
I=1
WHILE B[I]≠'\0'
PREZIME[I]=B[I]
I=I+1
ENDWHILE
PREZIME[I]='\0'
GU=C
GS=D
BPI=F
```

```
STAMP_STUD() FROM STUDENT
OUTPUT IME, PREZIME
OUTPUT GU, GS, BPI
OUTPUT PROS
```

```
DAJ_BPI(): INTEGER FROM STUDENT
RETURN BPI
```

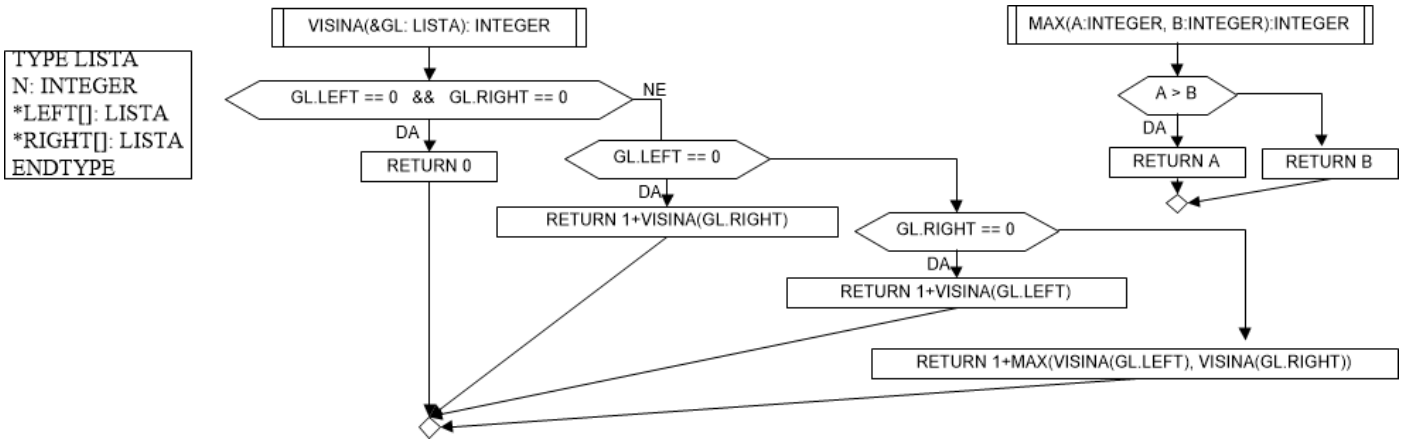
```
DAJ_PROS(): FLOAT FROM STUDENT
RETURN PROS
```

```
POV_GS() FROM STUDENT
GS=GS+1
```

```
AZURIRANJE(A: INTEGER) FROM STUDENT
PROSJEK=(PROSJEK*BPI+A) / (BPI+1)
BPI=BPI+1
```

```
PROVJERA(): INTEGER FROM STUDENT
IF (GU<1960 \/ GU>2006 \/ BPI<0 \/ BPI>40 \/ GS<1 \/ GS>5 \/
PROSJEK<6 \/ PROSJEK>10)
RETURN 0
ELSE
RETURN 1
ENDIF
```

3. Pretpostavljamo da je formirano binarno drvo i to preko struktura. Napišimo funkciju koja određuje visinu drveta. Funkciji se prosljeđuje pokazivač na korijen drveta.



4. Kreirati algoritamsku funkciju koja provjerava da li je graf zadat kao usmjeren ili neusmjeren. Podrazumijevati da je matrica susjedstva grafa zadata u glavnom dijelu algoritma i proslijeđena funkciji.
Napomena: Da bi graf bio neusmjeren njegova matrica susjedstva treba da bude simetrična, sa svim elementima na glavnoj dijagonali jednakim 0.

```
X[10][10]: INTEGER
N: INTEGER
INPUT N
I=1
WHILE I<=N
      J=1
      WHILE J<=N
            INPUT X[I][J]
            J=J+1
      ENDWHILE
      I=I+1
ENDWHILE

IF FUKNCIJA(X,N) ≡ 1
      OUTPUT "GRAF USMJEREN"
ELSE
      OUTPUT "GRAF NEUSMJEREN"
ENDIF
END


FUNKCIJA(&X[][]: INTEGER, N: INTEGER):INTEGER
I,J: INTEGER
I=1
WHILE I<=N
      J=1
      WHILE J<=N
            IF (I≡J AND X[I][J]≡1)
                  RETURN 0
            ENDIF
            IF (I≠J AND(X[I][J] ≠ X[J][I]))
                  RETURN 0
            ENDIF
            J=J+1
      ENDWHILE
      I=I+1
ENDWHILE
RETURN 1
```