1. Ukoliko se koristi statičko pravilo djelokruga, na aktivacijski zapis koje procedure pokazuje kazaljka nelokalnih imena potprograma C.

Glavni(){

def A(){

B();

}

def B(){

C();

}

def C(){

print("C");

}

A();

}

* 1. A
  2. **Glavni**
  3. B
  4. Ništa od navedenog

1. Označite produkciju koja zadovoljava uvjete L-atributne prijevodne gramatike u pojednostavljenom obliku pravila računanja vrijednosti svojstava.
   1. ⟨*X*⟩*h,i → aj* ⟨*Y*⟩*k* ⟨*Z*⟩*q,p* {*f*}*l,m,n,o*

*l ← h, m ← j, n ← k, q ← o, i ← p*

* 1. ⟨*X*⟩*i → aj* ⟨*Y*⟩*k* {*f*}*l,m,n,o* ⟨*Z*⟩*q,p*

*l ← k, m ← j, n ← k, q ← o, i ← p*

* 1. ⟨*X*⟩*h,i → aj* ⟨*Y*⟩*k* {*f*}*m,n,o* ⟨*Z*⟩*q,p*

*k ← h, m ← j, n ← k, q ← o, i ← p*

* 1. **⟨*X*⟩*h,i → aj* ⟨*Y*⟩*k* {*f*}*l,m,n,o* ⟨*Z*⟩*q,p***

***l ← h, m ← j, n ← k, q ← o, i ← p***

* 1. ⟨*X*⟩*h,i → aj* {*f*}*l,m,n,o* ⟨*Y*⟩*k* ⟨*Z*⟩*q,p*

*l ← h, m ← j, n ← k, q ← o, i ← p*

1. Generator ciljnog programa kao izlaz može imati različite ciljne jezike. Za premjestivi ciljni program vrijedi:
   1. Program povezivač izvodi postupak dorade adrese.
   2. Adrese naredbi su u potpunosti izrađene.
   3. Program punitelj povezuje zasebno prevedene procedure.
   4. Adrese podataka su u potpunosti izrađene.
   5. **Ništa od navedenog.**
2. Za zadani isječak programa koji broj mrežica će se odrediti primjenom algoritma bojanja grafova?

i = 1;

j = i + 1;

k = j + 1;

l = i + j + k;

i = 0;

1. **5**
2. 3
3. 4
4. 10
5. 6
6. Budući da se zahtjeva da preuređeni međukôd tijekom sinteze ciljnog programa sačuva svoje izvorno značenje, prije pretvorbe potrebno je analizirati izvođenje programa.

Analizu izvođenja programa čini poredani slijed analiza:

* 1. Analiza pseudonima, analiza tijeka izvođenja programa, analiza toka podataka, analiza zavisnosti podataka, postupci pretvorbe
  2. Analiza tijeka izvođenja programa, analiza pseudonima, analiza toka podataka, analiza zavisnosti podataka, postupci pretvorbe
  3. Analiza toka podataka, analiza pseudonima, analiza zavisnosti podataka, analiza tijeka izvođenja programa, postupci pretvorbe
  4. Odznači (ne želim odgovoriti)
  5. **Analiza tijeka izvođenja programa, analiza toka podataka, analiza zavisnosti podataka, analiza pseudonima, postupci pretvorbe**
  6. Analiza toka podataka, analiza tijeka izvođenja programa, analiza pseudonima, analiza zavisnosti podataka, postupci pretvorbe

1. Promatramo potisni automat parsera od vrha prema dnu za zadanu *L*-atributnu prijevodnu gramatiku.

Što se od navedenog ne nalazi na stogu tog automata na početku njegova rada?

* 1. Početni nezavršni znak gramatike
  2. Kazaljke koje pokazuju na mjesto zapisa vrijednosti izvedenih svojstava početnog nezavršnog znaka
  3. **Kazaljke koje pokazuju na mjesto zapisa vrijednosti nasljednih svojstava početnog nezavršnog znaka**
  4. Oznaka dna stoga
  5. Odznači (ne želim odgovoriti)
  6. Početne vrijednosti nasljednih svojstava početnog nezavršnog znaka

1. Parsiramo *L*-atributnu prijevodnu gramatiku metodom rekurzivnog spusta. Označite produkciju, ako takva postoji, kojoj je pridružen sljedeći potprogram.

S () {

Slučaj (Ulaz.znak) {

'while': {

Lokalne varijable x, y;

Ulaz.znak = sljedeći znak niza w;

ako (Ulaz.znak != '(')

Odbaci();

A(&x);

Ulaz.vrijednost = vrijednost sljedećeg znaka niza w;

ako (Ulaz.znak != ')')

Odbaci();

y = 1 - x;

B(y);

}

Svi ostali znakovi:

Odbaci();

}

}

* 1. *S → while(Ay)Bx*

*y ← 1 − x*

* 1. *Sx → while(Ax)By*

*y ← 1 − x*

* 1. *S → while(Ax)By*

*y ← 1 – x*

* 1. **Ništa od navedenog**
  2. *S → whilex(Ay)Bx*

*y ← 1 – x*

1. Koji od navedenih nije jezični procesor s obzirom na stupanj pripremljenosti ciljnog programa za izvođenje?
   1. **Ništa od navedenog**
   2. Generator izvodivog ciljnog programa
   3. Generator zasebnih dijelova programa
   4. Spremi-i-pokreni jezični procesor
   5. Odznači (ne želim odgovoriti)
   6. Generator premjestivog ciljnog programa
2. Obilježja se dodjeljuju različitim dijelovima izvornog programa, odnosno različitim leksičkim i sintaksnim cjelinama.

Naredba pridruživanja definira se, između ostalog, sljedećom produkcijom gramatike:

⟨*S*⟩ *→ IDN* :=⟨*E*⟩

Obilježjima naredbi pridružuju se dvije vrijednosti: Bez Pogreške i Pogreška.

Produkcijama gramatike dodaju se sljedeće semantičke akcije:

⟨*S*⟩*V1 → IDNV2* =⟨*E*⟩*V3*

{

ako (1)

2

inače

3

}

**Napomena**! Potrebno je odabrati ispravne vrijednosti za 1, 2 i 3.

* 1. 1. V2 == V3

2. V2 = BezPogreške

3. V3 = Pogreška

* 1. 1. V2 != V3

2. V3 = BezPogreške

3. V3 = Pogreška

* 1. 1. V2 == V3

2. V2 = Pogreška

3. V2 = BezPogreške

* 1. **1. V2 != V3**

**2. V1 = Pogreška**

**3. V1 = BezPogreške**

* 1. 1. V2 == V3

2. V1 = Pogreška

3. V1 = BezPogreške

1. Ako su leksičke jedinke opisane pravilima , i , koje će jedinke prepoznati analizator prilikom analize niza ?

**r1**: jedan/1

**r2**: dva/2

**r3**: (a|b|...|z)\*(0|1|...9)\*

* 1. r1, r2
  2. **r1, r3, r3**
  3. r1, r3, r2
  4. r1, r1, r2
  5. r1, r3

1. Neka je zadan sljedeći program:

1) i = j + 1;

2) ako ( i > 5 )

3) skoči L1

inače

{

4) n = j \* k;

5) m = n + 1;

}

6) L1: n = m / 2;

Kako izgleda graf zavisnosti podataka za zadani program?

* 1. A picture containing clock

     Description automatically generated
  2. A picture containing clock

     Description automatically generated
  3. A picture containing clock

     Description automatically generated
  4. **A picture containing text, clock

     Description automatically generated ✓**
  5. A picture containing clock

     Description automatically generated

1. Za zadani graf tijeka izvođenja, koja od ponuđenih tvrdnji dominacije je istinita?

Diagram

Description automatically generated

* 1. *dom*(Početni) = {}
  2. ***dom*(B4) = {Početni, B1, B2, B4}**
  3. *dom*(B2) = {B1, B2}
  4. *dom*(B3) = {Početni, B1}
  5. *dom*(B5) = {Početni, B1, B3, B4, B5}

1. Što će ispisati sljedeći program ako se parametri razmjenjuju primjenom mehanizma razmjene adresa?

int x = 0;

int y = 1;

void f(a, b) {

x = 7;

b = 2;

y = a + b;

print(a, b, x, y);

}

f(x, y);

print(x, y);

* 1. 0 1 0 1 0 1
  2. **7 9 7 9 7 9**
  3. 7 2 7 9 7 9
  4. 7 2 7 9 0 1
  5. 7 2 7 9 0 1

1. Parsiramo *L*-atributnu prijevodnu gramatiku potisnim automatom. Neka je zadana sljedeća produkcija

⟨*S*⟩*n1,i1 →* {*f*}*n2,i2 ax* ⟨*A*⟩*i3 by* {*h*}*n3,n4,i4* ⟨*B*⟩*n5,n6,n7*

*n2 ← n1 , n3 ← i2, n4 ← x,* (*i1,n5*) *← i4, n6 ← i3 n7 ← y*

gdje je ⟨*S*⟩ početni nezavršni znak gramatike. Neka su elementi stoga, počevši od vrha prema dnu, indeksirani s brojevima *0, 1, 2…* . Pod pretpostavkom da se na stogu nalazi početna konfiguracija potisnog automata, što će biti zapisano unutar stoga na indeksu broj *9* nakon jednog koraka rada potisnog automata u kojem pročitamo ulazni završni znak *a* i primijenimo zadanu produkciju?

* 1. *y. vrijednost*
  2. ***i4. vrijednost***
  3. ⟨*B*⟩
  4. *x. vrijednost*
  5. *i3. vrijednost*

1. Koliko čvorova ima graf tijeka izvođenja idućeg programa?

input(x)

x = x % 10

if x > 5:

print("Yes")

goto END

else:

print("No")

END:

x = 0

* 1. 2
  2. 1
  3. **6**
  4. 8

1. Koje troadresne naredbe generira sintaksnom upravljanjo generiranje međukoda za sljedeći izraz:

a = b \* (c + d) - e

Oznaka p\* se koristi za pomoćna imena.

* 1. p1 = b \* c;

p2 = p1 + d;

a = p2 - e;

* 1. **p1 = c + d;**

**p2 = b \* p1;**

**a = p2 - e;**

* 1. p1 = c + d;

p2 = p1 - e;

a = b \* p2;

* 1. p1 = b \* p1;

a = p1 - e;

* 1. p1 = c + d;

a = b \* p1 - e;

1. Zadana je gramatika s produkcijama:

*S → aABc; S → bBc; A → ε; A → cB; B → bB; B → a*

Čemu je jednak skup *PRIMIJENI*(*A → ε*) ?

* 1. {*a, b, c*}
  2. {*a, c*}
  3. {*ε*}
  4. {*c*}
  5. **{*a, b*}**

1. U metodi rekurzivnog spusta za *L*-atributnu prijevodnu gramatiku, znakovima gramatike pridružuju se \_\_\_\_\_\_\_\_\_\_\_\_, a svojstvima znakova pridružuju se \_\_\_\_\_\_\_\_\_\_\_\_\_.
   1. **Potprogrami, programske varijable**
   2. Programske varijable, potprogrami
   3. Znakovi stoga, pravila preslikavanja
   4. Pravila preslikavanja, znakovi stoga
   5. Znakovi stoga, programske varijable
2. Dinamička provjera vrijednosti obilježja izvodi se tijekom:
   1. Leksičke analize
   2. Sintaksne analize
   3. Semantičke analize
   4. Generiranje međukoda
   5. **Izvođenje ciljnog programa**
3. Listu unaprijednih adresa i listu unazadnih adresa koristi:
   1. **Generator ciljnog programa**
   2. Semantički analizator
   3. Ciljni program
   4. Sintaksni analizator
   5. Potisni automat
4. U metodi rekurzivnog spusta za *L*-atributnu prijevodnu gramatiku, ako je varijabli pridruženo nasljedno svojstvo, razmjenjuje se njezina \_\_\_\_\_\_\_\_\_\_, a ako je varijabli pridruženo izvedeno svojstvo, razmjenjuje se njezina \_\_\_\_\_\_\_\_\_\_.
   1. Adresa, vrijednost
   2. **Vrijednost, adresa**
   3. Adresa, adresa
   4. Vrijednost, vrijednost
   5. Ništa od navedenog (nema razmjene varijabli)
5. Što od navedenog nije jedan od osnovnih mehanizama razmjene ulazno/izlaznih parametara procedura:
   1. Razmjena vrijednosti
   2. Povratna razmjena vrijednosti
   3. Razmjena adrese
   4. **Povratna razmjena adrese**
   5. Razmjena imena
6. Prilikom generiranja ciljnog programa na temelju postfiksnog sustava oznaka, izravnanje sintaksnog stabla ostvaruje se primjenom:
   1. LR-parsera
   2. LL(1)-parsera
   3. Potisnog automata
   4. **Potisnog stoga**
   5. Stabla dominacije
7. Analizu pseudonima čine dva dijela:
   1. Pretraživač i poveznik pseudonima
   2. Pretraživač i konstruktor pseudonima
   3. **Skupljač i prenositelj pseudonima**
   4. Skupljač i prevoditelj pseudonima
   5. Konstruktor i destruktor pseudonima
8. Pri traženju slobodnog segmenta memorijskog prostora, pretpostavimo da nijedan slobodni segment pojedinačno nije dovoljno velik za traženu memoriju. Koji se od navedenih postupaka u tom slučaju najprije pokušava provesti?
   1. Pakiranje slobodnih segmenata
   2. Pakiranje zauzetih segmenata
   3. Združivanje susjednih zauzetih segmenata
   4. **Združivanje susjednih slobodnih segmenata**
   5. Združivanje svih slobodnih segmenata
9. Dio opisnika procedure po kojem se razlikuju statičko i dinamičko pravilo djelokruga jest:
   1. Statička memorija
   2. Upravljačka kazaljka
   3. Kazaljka stoga
   4. **Kazaljka nelokalnih imena**
   5. Kazaljka sata
10. U analizi dominacije koju provodimo nad grafom tijeka izvođenja programa *dom*(x) označava skup dominatora čvora x. Čvor d je neposredni dominator čvora *a* ako i samo ako za svaki drugi čvor *c* t.d. vrijedi:
11. Što od navedenog nije dio opisnika procedure:
    1. Lokalni podatci
    2. **Statička memorija**
    3. Upravljačka kazaljka
    4. Kazaljka nelokalnih imena
    5. Vrijednost ulaznih parametara
12. Izvođenje naredbi za koje vrijedi da se vrijednost relacije \_\_\_\_\_\_\_ ne mijenja određuje životni vijek pridruživanja imena
    1. Stanja
    2. **Okoline**
    3. Naslovljavanja
    4. Odnosa
    5. Zavisnosti
13. Strojno nezavisni program virtualnog stroja u strojni program prevodi:
    1. Jezični procesor
    2. **Jezični postprocesor**
    3. Program povezivač
    4. Program punitelj
    5. Jezični pretprocesor
14. Tijekom analize strukture pri analizi tijeka izvođenja programa, analiza strukture obilazi graf tijeka izvođenja programa, traži podgrafove uzorke, zamijeni ih jednim zamjenskim čvorom i gradi:
    1. Usmjereni graf izvornog programa
    2. Stablo čvorova
    3. Dominantno stablo
    4. Graf uzoraka
    5. **Upravljačko stablo**
15. Što od navedenog nije jedan od tipova jezičnih procesora u podjeli s obzirom na stupanj pripremljenosti ciljnog programa za izvođenje?
    1. Spremi-i-pokreni jezični procesor
    2. Generatori izvodivog ciljnog programa
    3. **Generatori produkcija strojnog jezika**
    4. Generatori premjestivog ciljnog programa
    5. Generatori zasebnih dijelova programa
16. Gradimo atributnu prijevodnu gramatiku koja generira troadresne naredbe za računanje logičkih izraza. Produkciju Eime1,kod1 -> ¬ Eime2,kod2 ima smisla proširiti sljedećim akcijskim znakovima:
    1. Ime1=novoIme(); Kod1=Generiraj(Ime1 || '':=not'' Ime2)
    2. Ime2=novoIme(); Kod2=Generiraj(kod1 || '':=not'' Ime2)
    3. Ime1=novoIme(); Kod1=Generiraj(kod1 || Ime1 '':=not'' Ime2)
    4. **Ime1=novoIme(); Kod1=Generiraj(kod2 || Ime1 '':=not'' Ime2)**
    5. Ime2=novoIme(); Kod2=Generiraj(Ime1 || '':=not'' Ime2)
17. Program Glavni sadrži procedure A(x), B(y), C(z) od kojih nijedna ne poziva samu sebe. Ne pozivaju se ni međusobno, osim procedure A koja u nekim slučajevima poziva B i C. Koja je najveća moguća dubina stabla aktiviranja procedura za program Glavni uz pretpostavku da je dubina korijena 1?
    1. 2
    2. **3**
    3. 4
    4. 5
    5. Neograničena
18. Pri optimiranju procedura međukoda srednje razine koristi se pretvorba rekurzivnih procedura u:
    1. **Petlje**
    2. Zasebne pozive za svaki parametar
    3. Procedure s jednim prijenosnim parametrom
    4. Rekurzivne procedure nije moguće preoblikovati
    5. Skokove
19. Koraci postupka analize izvornog programa redom su:
    1. Leksička, semantička i sintaksna analiza
    2. Sintaksna, semantička i leksička analiza
    3. **Leksička, sintaksna i semantička analiza**
    4. Semantička, sintaksna i leksička analiza
    5. Semantička, leksička i sintaksna analiza
20. Tijekom prevođenja izvornog programa pomoću Co-No tablice, trenutni operator „;“ i sljedeći operator „+“ određuju sljedeću akciju generatora ciljnog programa:
    1. Pomnoži
    2. **Dohvati**
    3. Zbroji
    4. Obriši
    5. Spremi
21. Pri izgradnji potisnog automata za atributnu prijevodnu gramatiku, temeljem produkcije , pri čemu su i izlazni znakovi, gradi se akcija:
22. Za sljedeću kontekstno neovisnu gramatiku odredite relaciju za završni znak *c*.
23. U LL(1) gramatici, za praznu produkciju , relacija izračunava se kao:
24. Dva su osnovna načina traženja slobodnog segmenta memorijskog prostora: potraga za prvim slobodnim segmentom dovoljne veličine i potraga za slobodnim segmentom koji je po svojoj veličini \_\_\_\_\_\_\_\_\_\_ tražene memorije.
    1. Jednak
    2. **Veći, ali najbliži veličini**
    3. Veći od
    4. Manji, ali najbliži veličini
    5. Manji od
25. Leksički analizator slijedno čita tekst izvornog programa:
    1. **Znak po znak**
    2. Riječ po riječ
    3. Liniju po liniju
    4. Blok po blok
    5. Funkciju po funkciju
26. Ako je LR stavka potpuna, onda je oznaka točke na \_\_\_\_\_\_\_\_\_\_ mjestu \_\_\_\_\_\_\_\_\_\_ strane produkcije.
    1. Krajnje lijevom, desne
    2. Krajnje desnom, lijeve
    3. **Krajnje desnom, desne**
    4. Proizvoljnom, lijeve
    5. Proizvoljnom, desne
27. Za sljedeću gramatiku odredite vrstu:
    1. Regularna gramatika
    2. Q-gramatika
    3. S-gramatika
    4. **LL(1) gramatika**
    5. Operaterska gramatika
28. Za Q-gramatiku s produkcijama odredite skup .
29. Koje upravljačke tablice koristi parser Pomakni-Reduciraj?
    1. Reduciraj
    2. Reduciraj, Odbaci
    3. Stavi, NovoStanje
    4. Pomakni, Odbaci
    5. **Stavi, Pomakni/Reduciraj**
30. Za ciljni program prikazan u donjem dijelu stranice dodjeljuju se registri temeljem Cockeovog algoritma, a za bojenje grafa koristi se Chaitinov algoritam. Koliko se stvarnih registara dodjeljuje ciljnom programu?

1)

2)

4)

5)

6)

7)

* 1. 1
  2. 2
  3. **3**
  4. 4
  5. 5

Za programski odsječak u donjem dijelu stranice

var i = 1

var b[3] = {5, 6, 7}

procedura f(x)

i = 0

x = 8

ispis(b[0], b[1], b[2])

i = 2

x = 9

kraj

f(b[i])

ispis(i, b[0], b[1], b[2])

odredite ispis ako se pri pozivu potprograma koristi:

1. razmjena vrijednosti:
   1. 5,6,7; 2,5,9,7
   2. **5,6,7; 2,5,6,7**
   3. 8,6,7; 1,8,6,9
   4. 5,6,7; 1,5,6,7
   5. 5,7,6; 2,5,9,7
2. povratna razmjena vrijednosti:
   1. 5,6,7; 2,5,6,7
   2. 8,6,7; 1,8,6,9
   3. **5,6,7; 2,5,9,7**
   4. 5,6,7; 1,5,9,7
   5. 5,8,7; 2,5,9,7
3. razmjena imena:
   1. 8,6,7; 1,8,6,9
   2. 5,6,7; 2,5,9,7
   3. 5,6,7; 1,5,6,7
   4. **8,6,7; 2,8,6,9**
   5. 5,8,7; 2,5,9,7
4. razmjena adresa:
   1. **5,8,7; 2,5,9,7**
   2. 5,6,7; 1,5,6,7
   3. 5,6,7; 2,5,9,7
   4. 8,6,7; 2,8,6,9
   5. 5,8,7; 1,5,9,7

Za program u donjem dijelu stranice

01 Glavni()

02 int a = 1;

03 X(x)

04 {

05 vrati x + 1;

06 }

07 Y(y)

08 {

09 vrati X(y \* a);

10 }

11 Z(z)

12 int a = 2;

13 {

14 vrati Y(z \* a);

15 }

16 {

17 X(5);

18 Y(5);

19 Z(5);

20 }

nacrtajte stablo aktiviranja procedura i za to stablo odredite:

1. Broj čvorova stabla:
   1. 3
   2. 4
   3. 6
   4. **7**
   5. 10
2. Parametar poziva procedure u čvoru najveće dubine stabla ako koristimo **statičko** pravilo djelokruga:
   1. 5
   2. **10**
   3. 11
   4. 20
   5. 21
3. Parametar poziva procedure u čvoru najveće dubine stabla ako koristimo **dinamičko** pravilo djelokruga:
   1. 5
   2. 10
   3. 11
   4. **20**
   5. 21
4. Generiranje ciljnog programa na temelju postfiksnog sustava oznaka: Ako se u međukodu pročita operator, onda generator primijeni akciju:
   1. stavi pročitani znak međukoda na vrh stoga, a glavu za čitanje zadrži na trenutnom znaku
   2. stavi pročitani znak međukoda na vrh stoga i pomakni glavu za čitanje na sljedeći znak
   3. generiraj naredbe ciljnog programa i pomakni glavu za čitanje na sljedeći znak
   4. **uzmi s vrha stoga zadani broj operanada, generiraj naredbe ciljnog programa i stavi rezultirajući operand na vrh stoga**
   5. uzmi s vrha stoga zadani broj operanada, generiraj naredbe ciljnog programa
5. Zadana je produkcija L-atributne prijevodne gramatike: gdje su te izvedena svojstva, a i su nasljedna svojstva. Nasljedno svojstvo može se računati na temelju svojstava:
6. Za sljedeću kontekstno neovisnu gramatiku izračunajte vrijednost relacije za završni znak *b*.
7. Ako je stanje parsera označeno stavkama i , parser će učitavanjem znaka *b* iz ulaznog niza izvesti akciju:
   1. neće moći jednoznačno odlučiti zbog proturječja
8. Tekst se odnosi na sljedeća dva zadatka. Za program naveden na dnu druge stranice

01 Glavni()

02 def X(a)

03 def Y(x)

04 {

05 vrati x\*3;

06 }

07 def Z(x)

08 {

09 vrati x%2;

10 }

11 {

12 vrati a\*a;

13 }

14 {

15 ispiši Y(4);

16 }

Navedite gdje pokazuje kazaljka nelokalnih imena procedure Y u trenutku izvođenja naredbe 05 ako se koristi:

(i) statičko pravilo djelokruga ugniježđenih procedura

* 1. **opisnik procedure Glavni**
  2. opisnik procedure X
  3. opisnik procedure Y
  4. opisnik procedure Z
  5. ništa od navedenog

1. (ii) dinamičko pravilo djelokruga ugniježđenih procedura
   1. **opisnik procedure Glavni**
   2. opisnik procedure X
   3. opisnik procedure Y
   4. opisnik procedure Z
   5. ništa od navedenog

Za programski odsječak prikazan na dnu stranice

var x = 0

var A[2] = {10, 20}

procedura P(a)

x = 1

a = 100

ispiši(A[0], A[1])

x = 0

a = 101

kraj

P(A[x])

Ispiši(x, A[0], A[1])

Odredite ispis ako se kod poziva potprograma koristi:

1. (i) razmjena vrijednosti
   1. **10,20; 0,10,20**
   2. 100,20; 0,101,20
   3. 10,100; 0,101,100
   4. 10,20; 0,101,20
   5. 10,20; 0,101,100
2. (ii) razmjena adresa
   1. 10,20; 0,10,20
   2. **100,20; 0,101,20**
   3. 10,100; 0,101,100
   4. 10,20; 0,101,20
   5. 10,20; 0,101,100
3. (iii) razmjena imena
   1. 10,20; 0,10,20
   2. 100,20; 0,101,20
   3. **10,100; 0,101,100**
   4. 10,20; 0,101,20
   5. 10,20; 0,101,100
4. (iv) povratna razmjena vrijednosti
   1. 10,20; 0,10,20
   2. 100,20; 0,101,20
   3. 10,100; 0,101,100
   4. **10,20; 0,101,20**
   5. 10,20; 0,101,100
5. Navedena leksička pravila opišite regularnim izrazom (pripadni regularni izraz dopišite pored/ispod svakog pravila):
   1. Ključna riječ *ako*: **ako**
   2. Imena identifikatora koja počinju znakom *a* ili *b* i nastavljaju se nizom znakova *a*, *b* i *c* proizvoljne duljine:
   3. Brojevne konstante koje se zapisuju u oktalnoj bazi pri čemu konstanta počinje znamenkom *0*: