(1.) Nabroji i ukratko opiši svaki od osnovnih koraka u izgradnji JP

Osnovni koraci: definicija, strukturiranje, programsko ostvarenje, ocjena i održavanje jezičnog procesora

Definicija – pri definiciji određuju se funkcijski zahtjevi procesora (pravila izvornog jezika, svojstva ciljnog jezika, veličina, brzina rada, cijena, ...)

Strukturiranje – određuju se osnovni koraci rada jezičnog procesora kao i njihova sučelja.
– završava opisom cjelokupne organizacije jezičnog procesora, te razradom svih koraka

Programsko ostvarenje – izbor programskog jezika, izbor programske okoline,
– gradnja i ispitivanje samog programa

Ocjena – "ocjenjivanje" svojstava → ispitivanje dosljednosti
– ispitivanje koliko JP zadovoljava svojstva zadana definicijom

Održavanje – unapređivanje svojstava, poboljšavanje, ispravljanje grešaka

(2.) Zadaci LA (barem 5 s kratkim opisom)

* izravno pristupa znakovima teksta izvornog programa spremljenog u memoriji računala

* kodiranje znakova → koristi jedan od standardnih kodova (npr ASCII)

→ znakove istog tipa kodira jednim kodom (npr. za sve znamenke koji su brojke)

* odbacuje sve znakove koji se ne koriste u daljnjim fazama rada jezičnog procesora

↳ izbacuje komentare, znakove za tekstualnu strukturu → bjeline (znak novog reda, tabulator, ..)

* određuje klase lex jedinki

↳ za svc klase definirajú se pravila pripadnosti, LA spaja jedinke s pravilom

* provjerava da li lex jedinka zadovoljava pravila klase u koji je svrstana

ako nije moguće smjestiti u nijednu klasu → ispisuje GREŠKU

* određuje mjesto pogreške u izvornom programu i opisuje pogrešku

↳ postupci povezani s oporavkom

(3.) Navesti i objasniti varijable koje koristi simulator zasnovan na tablici prijelaza DKA
↳ objasniti postupak simulatora za grupiranje i određivanje lex klase jedinke

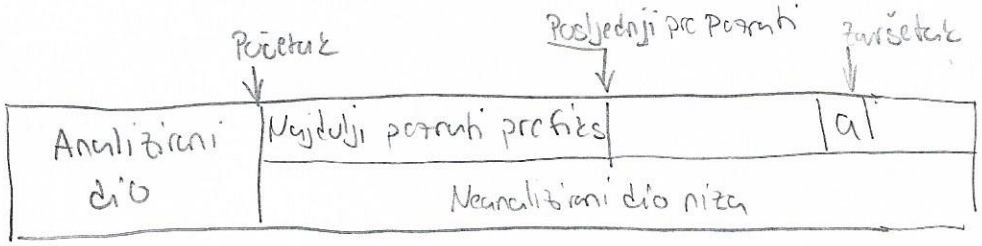
Koristi 4 sljedeće varijable:

Početak - kazaljka koja pokazuje na početak neanaliziranog dijela niza

Završetak - kazaljka koja pokazuje na posljednji pročitani znak

Posljednji - kazaljka koja pokazuje na posljednji znak najdužeg prepoznatog prefiksa niza

Izraz - poprima vrijednosti oznake



Završetak je fiksna kazaljka, početak se pomiče kada nakon uspješnog analiziranog dijela, čita se znak po znak do kraja ulaznog niza i Posljednji prepoznati pamti zadnji prepoznati niz. Ako ni jedan niz nije prepoznat → GREŠKA

4. Naredite barem 5 različitih vrsta oznaka za zapis sintaksnih pravila

Sintaksna pravila najčešće se definiraju:

* kontekstno neovisnom gramatikom
* regularnim izrazima
* BNF sustavom oznaka
* COBOL sustav oznaka

5. Nisam našao $\overset{||}{\wedge}$

(6) Napišite postupak sintaksne analize zasnovane na Co-No tablici

Co-No (Current operator - Next operator)

↳ jedan od najstarijih načina analize i prevođenja izvornog programa u ciljni program.

↳ prednost je brzina, a nedostatak neučinkovito korištenje memorije

Postupak analize izvornog programa i generiranja programa zadaje se dvodimenzionalnom tablicom veličine N x N

N - broj različitih operatora koji se koriste u izvornom programu

redni → lijevi operator stupci → desni operator

Ako je par operatora dozvoljen sintaksnim pravilima, onda element tablice koji je određen tim parom operatora označava jednu od akcija generatora ciljanog programa. U protivnom, u element tablice se upisuje greška.

Pr

	)	→
(	Greška	Dohvati
→	spremi	Greška

6. Napisati i objasniti na primjeru dva osnovna načina rješavanja nejednoznačnosti u LA

Dva načina: *pretraživanjem lijevog konteksta
*pretraživanje desnog konteksta

a) Pretraživanje desnog konteksta

r/r' → r i r' regularni izrazi

↳ niz u ulaznom spremniku grupira se u lex jedinku definiranu izrazom r ako i samo ako iza slijedi izraz r'.

Pr DO / (slovo + brojka)* = (slovo + brojka)*,

Niz je ključna riječ DO ako i samo ako slijedi niz definiran reg izrazom (slovo+brojka)* = (slovo + brojka)*,

Ako se iza ne nalazi zarez → DO je dio identifikatora a ne ključna riječ

6. b) Pretraživanje lijevog konteksta

↳ definiraju se dodatna stanja

↳ ulazak i izlazak definiraju se akcijama pridruženim pojedinim regularnim izrazima

Lijeva kontekstna ovisnost:

<Ime Stanja> r,

↳ označka da je niz definiran regularnim izrazom r ako i samo ako je simulator u dodatnom stanju ImeStanja

Pr: X!++

r_1	X	Ispiši ("r_1")
r_2	X(+!!)	Uđi u stanje S
		~~ODBACI~~
r_3	<S> !++	Ispiši ("r_3")
		Izađi iz S

X! -> uđi u stanje S

X -> "r_1"

+!! -> "r_3" -> izađi iz S

↳ Primjer s auditornih

(7) Napisati algoritam za računanje skupova PRIMIJENI

Algoritam računanja skupova PRIMIJENI:

1) Određivanje praznih nezaršnih znakova
2) Računanje relacije : započinje izravno znakom
3) Računanje relacije : započinje znakom
4) Računanje skupova ZAPOČINJE za nezavršne znakove
5) Računanje skupova ZAPOČINJE za produkcije
6.) Računanje relacije izravno ispred znaka
7.) Računanje relacije izravno kraj
8.) Računanje relacije kraj
9.) Računanje relacije ispred
10.) Proširenje relacije ispred oznakom kraja niza: ⊥
11.) Računanje skupova SLIJEDI za sve prazne nezaršne znakove
12.) Računanje skupova PRIMIJENI

(8.) Definirati relacije na temelju čijih uvjednosti se gradi tablica Pomakni/Pronađi

↳ tablica se gradi na temelju relacija: * Ispod znaka
* Reducira znakom

Relaciji Ispod znaka

↳ neka je zadana gramatika bez ε-produkcija i pratnih nezavršnih znakova

↳ završni ili nezavršni znak A je znak stoga

↳ završni znak X je znak ulaznog niza

Vrijedi relacija:

Ispod znaka (A, X)

ako i samo ako je ispunjen jedan od sljedeća 2 uvjeta:

1) A je izravno ispred B (znak) na desnoj strani barem jedne produkcije zadane gramatike, odnosno vrijedi relacija IzravnoIspred(A, B), a znak X započinje barem jedan niz generiran iz znaka B, odnosno X ∈ ZAPOČINJE (B)

2) A je oznaka dna stoga ▽ i X ∈ ZAPOČINJE (<S>) gdje je <S> početni nezavršni znak gramatike

Relacija Reducirantznakom

↳isti uvjeti ko i prije A→ znak stoga X→ jedni niz

Vrijedi relacija:

Reducirantznakom(A,X)

ako i samo ako je ispunjen jedan od sljedeća dva uvjeta:

1) A je krajnje desni znak strane produkcije $\langle L \rangle \Rightarrow L A$,
   a X slijedi znak $\langle L \rangle$ u barem jednom nizu generiranom
   iz početnog nezavršnog znaka $\langle S \rangle$, odnosno vrijedi
   $X \in$ SLIJEDI($\langle L \rangle$)

2) A je početni nezaršni znak gramatike $\langle S \rangle$,
   a X je oznaka kraja niza $\perp$

(9.) Opisati tri najčešće primjenjiva formalna modela semantičkog analizatora

Zasnivaju se na: prevođenju jezika, simulaciji na apstraktnom stroju i skupu aksioma

* prevođenje izvornog jezika
  - ↳ najjednostavniji formalni model semantičkog analizatora
  - ↳ bima definiranu semantiku

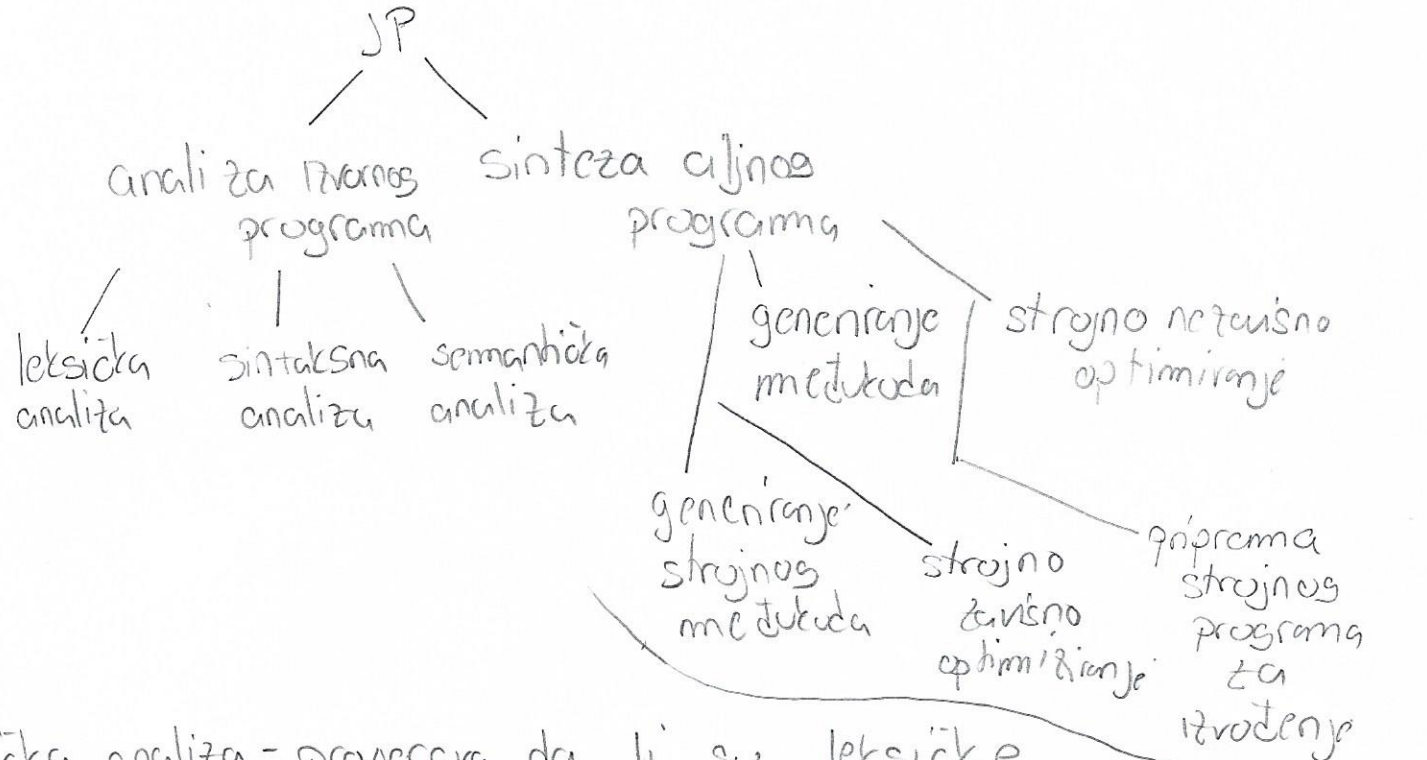
* izvođenje na apstraktnom stroju
  - ↳ zasniva se na skupu pravila primjenom kojih se simulira izvođenje izvornog programa
  - ↳ apstraktno računalo definira se stanjima, a izvođenje programa simulira se skupom funkcija koje mijenjaju stanja

* skup aksioma
  - ↳ aksiomi → logičke tvrdnje koje zadaje korisnik u različitim dijelovima izvornog programa → ističu očekivani rezultat tog dijela programa
  - ↳ semantička ispravnost zasniva se na provjeri istovjetnosti statičkih logičkih tvrdnji i rezultata dinamičkog izvođenja programa
  - ↳ nedostatak → opterećenje korisnika (zadaje naredbe izvornog programa koji definira način ispitivanja, a primjenom aksioma definira

(10) Opisati koracke gradnje atributivnog generativnog stabla.
Što je potpuno generativno stablo?

Potpuno g

(11.) Navesti sve korake rada JP i grupirati ih u dvije osnovne faze



Leksička analiza - provjerava da li su leksičke jedinke u izvornom programu pravilno napisane
- prevodi u niz parova (klasaLeksičkeJedinke, LexJedinka)
  ↳pr (KR, ako)

Sintaksna analiza - lex jedinke koje generira lex analizator grupira u hijerarhijske skupine sa zajedničkim značenjem (najčešće opisana sintaksnim stablom

-