Örebro universitet
Institutionen för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i

Kompilatorer och interpretatorer

för Dataingenjörsprogrammet m fl

tisdag 21 augusti 2018

Gäller som tentamen för:
DT125G Kompilatorer och interpretatorer, provkod 0100
DT3030 Datateknik C, Kompilatorer och interpretatorer, provkod 0100

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 38 För godkänt betyg krävs totalt minst 21 poäng, varav minst 8 poäng på uppgift 1.
Resultat:	Meddelas på kursens hemsida eller via e-post senast tisdag 11 september 2018.
Återlämning av tentor:	Elektroniskt via Studentforum.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070 - 73 47 013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara och tvetydiga formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

Formelsamling

1. Eliminering av vänsterrekursion

En vänsterrekursiv grammatik kan skrivas om så att den inte är vänsterrekursiv. Antag att en regel (eller, korrektare uttryckt, två produktioner) i grammatiken ser ut så här:

```
A -> A x | y
```

A är en icke-terminal, men **x** och **y** står för godtyckliga konstruktioner som består av terminaler och icke-terminaler.

Regeln ersätts av följande två regler (eller, korrektare uttryckt, tre produktioner), som beskriver samma språk men som inte är vänsterrekursiva:

```
A -> y R
R -> x R | empty
```

2. Vänsterfaktorisering

Antag att grammatiken innehåller denna regel (två produktioner):

```
A -> x y | x z
```

A är en icke-terminal, men **x**, **y** och **z** står för godtyckliga konstruktioner som består av terminaler och icke-terminaler.

Skriv om till dessa tre produktioner:

```
A -> x R
R -> y | z
```

Uppgift 1 (10 p)

När man beskriver hur kompilatorer fungerar brukar man prata om faser.

a) Vad är en fas?

b) Ange vilka faser det är, och beskriv kort vad varje fas är.

Uppgift 2 (5 p)

När vi kompilerar följande försök till C-program, får vi de kursiverade fel- och varningsmeddelandena. I vilka faser, eller vid andra tidpunkter, upptäcks vart och ett av de olika felen och varningarna?

```
include <stdio.h> // error: expected another token before '<'
#includddd <string.h> // error: invalid preprocessing directive

int main(void) {
    string s; // error: unknown type name 'string'
    printf("Hej!); // error: missing terminating " character
    int i;
    scanf("%d", i); // warning: format '%d' expects pointer
    return "OK"; // warning: return makes integer from pointer
}
```

Uppgift 3 (5 p)

Det här är ett programavsnitt i ett C-liknande språk:

```
x = 10;
while (x != y) {
    z = z + 20;
    if (x - y - z < 30)
        x = (x + y) / 40;
    else
        z = 50;
}
y = 60;
```

Översätt ovanstående programavsnitt till *två* av följande tre typer av mellankod. (Skulle du svara på alla tre, räknas den med högst poäng bort.)

a) ett abstrakt syntaxträd (genom att rita upp trädet!)

b) postfixkod för en stackmaskin

c) treadresskod

Scenario till de följande uppgifterna

Här är en grantopp med några kottar:

Vi ska lagra data om träd och kottar. En kotte har en längd och en vikt. Ett träd har en höjd, och högst tusen kottar.

Vi ska skapa ett språk som beskriver en skog. Skogen innehåller noll eller flera träd, och varje träd har noll eller flera kottar. Här är ett exempel på en inmatning i det språket:

```
skog
    träd 14.23
        kotte 14.2 0.12
        kotte 13 0.11
    slut träd
    träd 11
    slut träd
slut skog
```

Detta beskriver en skog med två träd: ett träd som är 14.23 meter högt och har två kottar, och ett träd utan några kottar. De två kottarna är 14.2 respektive 13 centimeter långa, och väger 0.12 respektive 0.11 kilo.

Inmatningen ska kunna anges på fritt format, så den här inmatningen är ekvivalent med den ovanstående:

```
                skog träd        14.23 kotte
14.2 0.12 kotte 13 0.11 slut
```

```
träd träd 11 slut träd slut

          skog
```

Uppgift 4 (2 p)

Vilka terminaler, dvs typer av tokens, behövs för att man ska kunna skriva en grammatik för inmatningsspråket i scenariot?

Uppgift 5 (4 p)

Skriv en grammatik för ett inmatningsspråket. Startsymbolen ska vara **inmatning**, som representerar en skog, enligt scenariot ovan.

Uppgift 6 (4 p)

a) Ett parse-träd (ibland kallat "konkret syntaxträd") innehåller noder för alla icke-terminaler. Rita upp parse-trädet för det första exemplet i scenariot.

b) Det står i scenariot att inmatningen ska kunna anges på fritt format, och det visas också en alternativt formaterad inmatning. Hur skiljer sig parse-trädet för denna andra inmatning från parse-trädet för den första?

Uppgift 7 (8 p)

Skriv en prediktiv recursive-descent-parser för språket i scenariot, i ett programspråk som åtminstone liknar C, C++, C# eller Java. Du behöver inte skriva exakt korrekt programkod, men det ska framgå vilka procedurer som finns, hur de anropar varandra, och vilka jämförelser med tokentyper som görs. Du kan anta att det finns en funktion som heter **scan**, som returnerar typen på nästa token, och en funktion som heter **error**, som man kan anropa när något gått fel och som skriver ut ett felmeddelande och avslutar programmet.
