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

Tentamen i

Kompilatorer och interpretatorer

för Dataingenjörsprogrammet m fl

måndag 29 oktober 2018

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

→ This exam is also available in an English version.

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 36. För godkänt betyg krävs totalt minst 18 poäng.
Resultat:	Meddelas på kursens hemsida eller via e-post senast måndag 19 november 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)

En kompilators arbete brukar delas in i ett antal faser. Ange vilka det är, och förklara kort vad varje fas gör. Vad är in- och utdata från respektive fas?

Uppgift 2 (8 p)

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

```
if (a < b) {
    while (c > d) {
        b = a + 1;
        a = c * (b + c);
        a = a + 1;
        d = c * (b + c);
    }
    b = a + 1;
}
c = a + 1;
```

a) Översätt ovanstående programavsnitt till *antingen* ett abstrakt syntaxträd (genom att rita upp trädet) *eller* postfixkod för en stackmaskin. (Inte båda!)

b) Översätt programavsnittet till treadresskod. Identifiera vilka basic blocks som finns, och rita upp flödesgrafen. (Flödesgrafen, med treadresskoden i, räcker som svar.)

c) Visa någon optimering som går att göra inom ett av dessa basic blocks.

Scenario till uppgift 3-6

Snart är det Halloween, och barnen går en "godisrunda", där de går runt till grannarna och frågar "bus eller godis". För att dokumentera sina aktiviteter behöver de ett särskilt Halloween-språk, där en inmatning kan se ut så här:

```
började godisrundan;
godis: tre karameller;
bus: gjorde ruskiga ljud;
godis: ett kilo choklad;
godis: en morot;
bus: eldade upp grannens bil;
gick hem;
```

Inmatningen beskriver en godisrunda. Den börjar med **började godisrundan;** och slutar med **gick hem;**. Däremellan anger man noll eller flera noteringar om **bus** eller **godis**. En sådan notering börjar med antingen **bus** eller **godis**, ett kolon (**:**), ett eller flera ord som anger vad man gjorde, och ett semikolon (**;**).

Inmatningen ska kunna skrivas på fritt format, som de flesta vanliga programmeringsspråk, till exempel så här:

```
började godisrundan ; godis : tre karameller ; bus
: gjorde ruskiga
   ljud ; godis : ett kilo choklad
; godis
: en morot ; bus : eldade upp
grannens bil ; gick hem ;
```

Uppgift 3 (3 p)

a) (1p) En av terminalerna, dvs typer av tokens, som behövs för att man ska kunna skriva en grammatik för halloween-språket är **ord**. Det är helt enkelt en eller flera små bokstäver, **a** till **z**. (Vi nöjer oss alltså med det engelska alfabetet.) Skriv ett reguljärt uttryck som beskriver hur ett ord får se ut.

b) (2p) Vilka andra terminaler, förutom **ord**, behövs för att man ska kunna skriva en grammatik för språket?

Uppgift 4 (4 p)

Skriv en grammatik för halloween-språket. Startsymbolen ska vara **godisrunda**, som representerar en inmatning enligt scenariot ovan.

Uppgift 5 (3 p)

Ett parse-träd (ibland kallat "konkret syntaxträd") innehåller noder för alla icke-terminaler. Rita upp parse-trädet för den här godisrundan, enligt din grammatik i uppgiften ovan:

```
började godisrundan;
godis: ett tuggummi;
gick hem;
```

Uppgift 6 (8 p)

Skriv en prediktiv recursive-descent-parser för halloween-språket, i ett språk 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.
