# Kompilatorer och interpretatorer för civilingenjörer: Hemtentamen 2021-01-13

Det här är hemtentan som går onsdag 13 januari 2021 i kursen DT501A Kompilatorer och interpretatorer för civilingenjörer, provkod A001. Ansvarig lärare är Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), telefon **070-73 47 013**.

Tid: 14:15 - 19:15

## Instruktioner

1. Den här hemtentan ersätter den planerade salstentan.
2. Uppgifterna ska lösas enskilt, dvs inga grupper av två eller flera studenter.
3. **Du får använda dator, böcker och vilka andra hjälpmedel som helst,** men du får inte samarbeta eller fråga någon (utom mig). Exempelvis är det tillåtet att söka och läsa på webbplatser som Stack Overflow, men inte att ställa egna frågor. Du kan alltså provköra kommandon och program, om du vill.
4. **Diskutera inte uppgifterna eller dela med dig av svar förrän tidigast dagen efter tentan.**
5. Lös de angivna uppgifterna och samla svaren på lämpligt sätt, till exempel i en PDF. Skicka sen in lösningarna till mig i Blackboard. (Gå till kursens sida i Blackboard, gå in i mappen **Hemtentor**, och välj den rätta tentan. Där kan man sedan välja att skicka in sin lösning. Då blir bedömningen anonym.)
6. Om det skulle bli problem med Blackboard, kan man i nödfall skicka in svaren antingen som ett kursmeddelande i Blackboard eller via vanlig e-post (thomas.padron-mccarthy@oru.se), senast vid tentatidens slut. Då blir bedömningen inte anonym.
7. I Blackboard ser du att du skickat in din lösning. Om du i stället skickade in med e-post, och inte senast en timme efter tentatidens slut fått ett svar från mig med en bekräftelse på att du skickat in svaren, bör du kontakta mig, enklast genom att ringa eller SMS:a mig (ifall det är e-posten som inte fungerar). Tänk på att en del mailtjänster (särskilt Microsoft-tjänster som Hotmail.com, Outlook.com och Live.com) ibland kastar bort brev med bilagor, utan att meddela det.
8. Skriv gärna svaren i ett ordbehandlingsprogram. Rita gärna eventuella diagram i ett ritprogram. Det är inte förbjudet att skriva och rita för hand, men då måste text och bilder scannas in eller fotograferas. Det finnas scanner-appar till Android och iPhone, till exempel Adobe Scan, som ger bättre resultat än att bara ta vanliga kort med kameran.
9. Tentatiden är utökad med en extra timme för att täcka in problem med e-post, inscanning eller fotografering av diagram, och liknande.
10. Om du behöver fråga något, så kontakta gärna mig. Ring eller skicka SMS, för jag kanske inte kommer att sitta vid datorn hela tiden.
11. Oklara och tvetydiga formuleringar kommer att misstolkas. Lösningar som inte går att läsa eller förstå kan naturligtvis inte ge några poäng.
12. Antaganden utöver de som står i uppgifterna måste anges. Gjorda antaganden

får inte förändra den givna uppgiften.
13. 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.
14. Maximal poäng är 30. För godkänt betyg krävs minst 16 poäng.
15. Resultat meddelas senast 15 arbetsdagar efter tentamensdatum. Eftersom svaren skickas in elektroniskt scannas tentorna inte för retur.

# 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
```

# Scenario till uppgifterna

Just nu är det aktuellt med vaccinationer. Det är viktigt att hålla reda på vilka som ska vaccineras, och vilka som redan är vaccinerade. Därför har vi skapat ett särskilt vaccinationsspråk.

Språket innehåller kommandona **person**, **prioritet**, **vaccinerat**, **lista** och **avsluta**:

- Kommandot **person** lägger till en eller flera personer som ska vaccineras. Man anger personnummer för personerna.
- Kommandot **prioritet** används för att ge prioritet till en eller flera personer. Prioriteten är ett heltal större än eller lika med noll. Personer som inte fått någon prioritet angiven har prioritet noll. Man skriver först prioriteten, och därefter personnumren på personerna det gäller.
- Kommandot **vaccinerat** används för att ange att en eller flera personer blivit vaccinerade.
- Kommandot **lista** skriver ut personnumren på personer. Man kan skriva **lista alla**, som listar alla personer, eller **lista vaccinerade**, som listar alla som redan är vaccinerade, i den ordning som de vaccinerades, eller **lista ovaccinerade**, som listar alla som inte är vaccinerade än, i prioritetsordning.
- Kommandot **avsluta** avslutar hela inmatningen.

Varje kommando skrivs in på en enda rad, och avslutas alltså med radslut, men inom raden kan man skriva på fritt format. Här är ett exempel på en komplett inmatning:

```
person 650402-7266
person 720515-4524 871031-1922
lista ovaccinerade
vaccinerat 871031-1922
prioritet 1 720515-4524
person 990101-3491
prioritet 100 650402-7266 720515-4524
 vaccinerat      720515-4524 650402-7266     990101-3491
lista     alla
avsluta
```

# Uppgift 1 (4 p)

a) Två av terminalerna som behövs för att man ska kunna skriva en grammatik för språket är **heltal** och **personnummer**. Ett heltal är ett positivt heltal skrivet med vanliga decimala siffror 0-9. Ett personnummer skrivs med tio siffror på det vanliga formatet. Skriv reguljära uttryck som beskriver hur heltal och personnummer får se ut.

b) Vilka andra terminaler, förutom **heltal** och **personnummer** behövs för att man ska kunna skriva en grammatik för språket?

# Uppgift 2 (4 p)

Skriv en scanner som kan dela upp inmatningen i tokens. Du får själv välja om du ska använda Flex eller skriva den för hand i C, C++ eller ett liknande språk. Scannern ska bestå av en funktion som returnerar en token-kod. Du kan anta att token-koderna finns definierade i form av makron eller enum-konstanter, ungefär som Bison gör när man har en Bison-grammatik med **%token**-deklarationer.

# Uppgift 3 (5 p)

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

# Uppgift 4 (4 p)

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

```
person 650402-7266 720515-4524
vaccinerat 871031-1922
avsluta
```

# Uppgift 5 (3 p)

Vaccinationsprogrammet i scenariot har en inmatning i form av ett särskilt språk, och på det viset liknar det en kompilator. Vi har ju en scanner och en parser. Skulle det vara rimligt att också ha en semantisk fas? Om nej, varför inte? Om ja, vad skulle den i så fall göra?

# Uppgift 6 (2 p)

En annan viktig del av en kompilator brukar vara symboltabellen. Om man tänker sig att vi bygger klart vaccinationsprogrammet, så det fungerar med utskrifter och allt, behöver programmet hålla reda på inmatade personnummer. Tabellen med personnumren kan motsvara symboltabellen. Vilka likheter och skillnader finns mellan den här personnummertabellen och en normal symboltabell i en kompilator?

# Uppgift 7 (8 p)

Skriv en prediktiv recursive-descent-parser för 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. Du kan anta att token-koderna finns definierade i form av makron eller enum-konstanter.

Om grammatiken från uppgift 3 ovan inte lämpar sig för en prediktiv recursive-descent-parser, måste den först göras om.

---

Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), 11 januari 2021