Kompilatorer och interpretatorer: Hemtentamen 2020-10-27

Det här är hemtentan som går tisdag 27 oktober 2020 i kursen DT135G Kompilatorer och interpretatorer, provkod A001. Den gäller även för den gamla kurskoden, DT125G. Ansvarig lärare är Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), telefon **070-73 47 013**.

Tid: 09:15 - 13:15

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 kursinnehållet, **Course Content**, 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. Skulle något krångla, med Blackboard eller vad det nu kan vara, är det fortfarande möjligt att skicka in lösningarna efter sluttiden, men då bör man ge någon form av förklaring till det.
7. Om det inte alls går att skicka in lösningarna som avsett i 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.
8. 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.
9. 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 finns

scanner-appar till Android och iPhone, till exempel Adobe Scan, som ger bättre resultat än att bara ta vanliga kort med kameran.
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 40. För godkänt betyg krävs minst 22 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
```

Uppgift 1 (5 p)

Vi ska göra ett gissa-spel. Spelaren ska tänka på ett tal, och datorn gissar vilket tal det är. Ett körexempel, med användarens inmatning understruken:

```
Tänk på ett tal och tryck ENTER.

Är det 7?
nej
Är det 7?
nej
Är det 7?
va?
Svara ja eller nej!
Är det 7?
sluta
Svara ja eller nej!
Är det 7?
nej
Är det 7?
ja
Jag vann!
```

Datorn gissar alltså hela tiden på talet 7, tills användaren ger upp och svarar "ja"!

Vi skriver spelet i form av ett C-program, men det blev tyvärr inte helt korrekt, och när vi försöker bygga och köra det får vi de angivna fel- och varningsmeddelandena. I vilka av kompilatorns faser, eller vid andra tidpunkter, upptäcks vart och ett av de olika felen och varningarna?

(Man kan behöva åtgärda en del av felen för att komma vidare och få de andra felen.)

Programmet	Meddelanden
`#include <stdio.h>`	
`int question(void) {`	
`    char svar[10];`	
`    while (true) {`	`'true' undeclared`
`        gets(svar);`	
`        if (strcmp(svar, "ja") == 0)`	`implicit declaration of 'strcmp'`
`            return 1; @`	`stray '@' in program`
`        else if (strcmp(svar, nej) == 0)`	`'nej' undeclared`
`            return 0;`	
`        else {`	
`            printf("Svara ja eller nej!\n");`	
`            return;`	`'return' with no value`
`        }`	
`    }`	
`}`	
`int Main(void) {`	`undefined reference to 'main'`

`    printf("Tänk på ett tal\n");`	
`    printf("och tryck ENTER.\n");`	
`    while (getchar() != "\n")`	`comparison between pointer and integer`
`        ;`	
`    do {`	
`        printf("Är det 7?\n");`	
`    } while (question() != 1e);`	`exponent has no digits`
`    printf("Jag vann!\n")`	`expected ';' before '}' token`
`} /* main`	`unterminated comment`

Uppgift 2 (5 p)

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

```
if (a - b < c * d * e) {
    while (a < b) {
        c = d * e + f;
        d = e * f * g;
    }
}
else {
    a = b;
}
```

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

Uppgift 3 (5 p)

I uppgiften ovan finns en while-loop med villkoret **a < b**, men ingen av variablerna **a** och **b** ändras i loopen. Om villkoret är sant från början, får man alltså en oändlig loop.

Förklara hur detta påverkar var och en av faserna i kompilatorn!

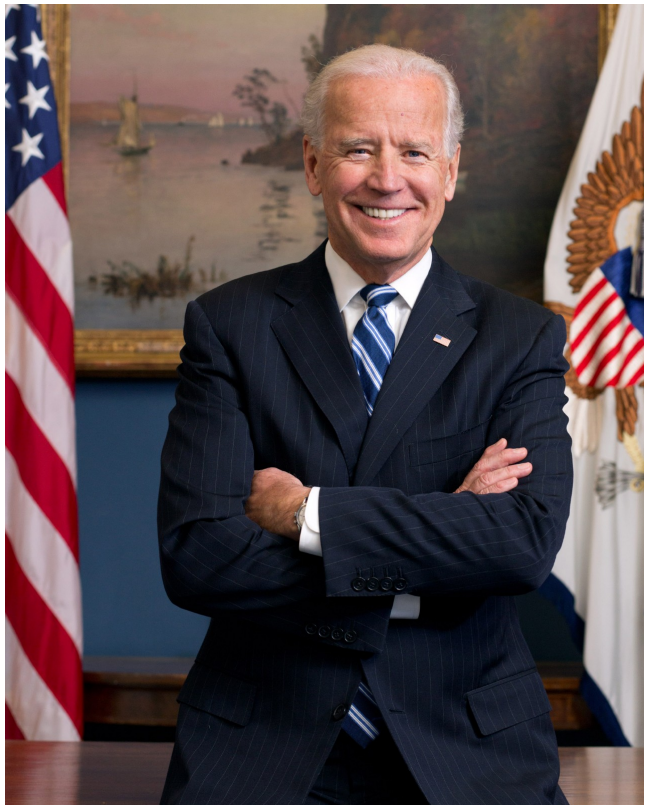
Uppgift 4 (2 p)

a) Vad är ett basic block?

b) Definitionen av basic block innehåller en del begränsningar av vilka hopp som är tillåtna. Varför finns dessa begränsningar?

Scenario till uppgift 5-8

I presidentvalet i USA nästa vecka står republikanernas kandidat **Donald Trump** mot demokraternas kandidat **Joe Biden**:

Förutom de två mest kända kandidaterna deltar inte mindre än 1220 andra kandidater, från **Aaron Avouris** (libertarian) till **Zoltan Gyurko Istvan** (republikan).

För att underlätta för rösträknarna ska vi göra ett program där man kan mata in kandidater och röster. Egentligen har USA ett komplicerat system där de olika delstaterna väljer så kallade elektorer, som i sin tur väljer presidenten, men vi förenklar lite. Här är ett exempel på inmatning:

```
candidate Joe Biden
vote Joe Biden
candidate Donald Trump
alias trump = Donald Trump
candidate Zoltan Gyurko Istvan
vote trump
vote Donald Trump
alias biden = Joe Biden
vote Joe Biden
vote Zoltan Gyurko Istvan
alias zoltan = Zoltan Gyurko Istvan
vote zoltan
vote Zoltan Gyurko Istvan
result
```

Inmatningen består av rader, och en rad kan vara:

- Ett kandidat-kommando, där man anger en kandidat genom att skriva **candidate** och sen en följd av ord som utgör kandidatens namn.
- Ett alias-kommando, där man anger ett alias för en kandidat genom att skriva **alias**, sen ett ord, sen ett likhetstecken, och sen namnet på en kandidat.
- Ett röstningskommando, där man skriver **vote** följt av antingen ett alias eller det fullständiga namnet på en kandidat.
- Ett resultat-kommando, **result**, som avslutar inmatningen.

Med inmatningen i exemplet ovan har **Zoltan Gyurko Istvan** vunnit med tre röster, före **Donald Trump** och **Joe Biden** som fick två röster var.

Varje kommando måste skrivas på en rad, men man kan ha godtyckliga blanktecken mellan orden.

Uppgift 5 (7 p)

a) En av terminalerna som behövs för att man ska kunna skriva en grammatik för språket är **ord**. Ett ord är en följd av stora eller små bokstäver. Skriv ett reguljärt uttryck som beskriver hur ett ord får se ut.

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

c) 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, ungefär som Bison gör när man har en Bison-grammatik med **%token**-deklarationer.

Uppgift 6 (5 p)

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

Uppgift 7 (3 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:

```
candidate Kim Jong Un
vote Kim Jong Un
result
```

Uppgift 8 (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.

Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se), 26 oktober 2020