Örebro universitet
Institutionen för teknik
Thomas Padron-McCarthy (Thomas.Padron-McCarthy@oru.se)

Tentamen i

Kompilatorer och interpretatorer

för Dataingenjörsprogrammet m fl

lördag 19 december 2009

Gäller som tentamen för:
DT3004 Datateknik C, Kompilatorer och interpretatorer, provkod 0100
TDK104 Kompilatorer och interpretatorer, provkod 0100

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 37. För godkänt betyg (3 respektive G) krävs 18 poäng.
Resultat:	Meddelas på kursens hemsida eller via e-post senast lördag 9 januari 2010.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på institutionen. Man kan också få sin rättade tenta hemskickad.
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!

Uppgift 1: Faser (5 p)

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

Uppgift 2: Scanning och reguljära uttryck (5 p)

a) (3p) Skriv reguljära uttryck för följande olika typer av tokens som förekommer i vanliga programmeringsspråk:

- teckenlösa heltal (alltså heltal som inte är negativa)
- heltal (som även kan vara negativa)
- identifierare, som kan innehålla siffror och bokstäver, men måste börja på en bokstav

b) (2p) Vad är det för skillnad på ett lexem och ett lexikaliskt värde, och vad har dessa med scanning att göra?

Uppgift 3: Grammatiker och top-down-parsning (15 p)

I följande grammatik är S, X och Y icketerminaler. **a**, **b**, **c**, **d** och **e** är terminaler. S är startsymbol.

```
S -> a X | b | X Y
X -> X c | e
Y -> d Y | d
```

a) Ange tio olika strängar som ingår i det språk som beskrivs av grammatiken ovan.

b) Grammatiken lämpar sig inte för implementation av en prediktiv recursive-descent-parser (engelska: *predictive recursive-descent parser*). Ange *vilka* problem det är, *vad* i grammatiken som orsakar problemen, och *vad som händer* i parsern.

c) Använd de transformationer som tagits upp i kursen för att omvandla grammatiken till en grammatik som beskriver samma språk, men som enkelt kan implementeras i en prediktiv recursive-descent-parser. Ange de generella transformationsregler som du använder, och visa vad resultatet blir för den här grammatiken.

d) Skriv den prediktiva recursive-descent-parsern i ett språk som åtminstone liknar C, Java eller Pascal. 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**, och att den returnerar typen på nästa token.

Uppgift 4: Mellankod (5 p)

```
x = 1;
while (y < 2) {
    if (z > 3) {
        t = t - x - y / (1 - z - y);
    }
    else {
        x = 1;
        y = 2;
        z = 3;
    }
}
```

Översätt ovanstående programavsnitt till *två* av följande tre typer av mellankod.

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

b) postfixkod för en stackmaskin

c) treadresskod

Observera: Det finns tre deluppgifter i uppgiften ovan. Välj ut och besvara (högst) *två* av dessa. (Skulle du svara på alla tre, räknas den med högst poäng bort.)

Uppgift 5: Några termer (7 p)

Förklara kort vad följande begrepp från kompilatortekniken innebär:

a) aktiveringspost
b) basic block
c) copy propagation
d) fas (engelska: *phase*)
e) pass (engelska: *pass*)
f) registerallokering
g) rotmängd (engelska: *root set*)
