Tentamen

Programspråksteori (5p)

21 mars 1998

Skrivtid: 9.00-15.00. Inga hjälpmedel
Totalpoäng: 40 (Godkänt 20)

Uppgift 1. (3p)
Visa för de nedanstående typdeklarationerna i Pascal vilken mängd av värden de representerar med hjälp av begreppen kartesisk produkt, disjunkt union och avbildning.

```
type    Color = (red, green, blue, yellow);
        Pixel = array [Color] of 0..256;
        Dot = record
                x : 0..480;
                y : 0..320;
                case hasColor : Boolean of
                        true : (c : Pixel);
                        false : ()
        end;
```

Uppgift 2. (4p)
Vad kännetecknar en deklaration? Beskriv de grundläggande formerna av deklarationer.

Uppgift 3. (3p)
Förklara vad det innebär att skapa abstraktioner över deklarationer och ge exempel på hur detta stöds i några programspråk.

Uppgift 4. (3p)
a) Givet följande programskelett, visa hur anropsstacken ser ut efter anropet av test, om man har statisk räckviddsbindning, visa både statiska och dynamiska länkar.
b) När A anropas, vad skrivs ut i proceduren test och i proceduren A, om man använder sig av statisk respektive dynamisk räckviddsbindning?

```
procedure A;
    integer i; real x, y;
    procedure test (a, b: integer);
        integer i;
    begin
     i:=3;
     x:=float(i)*y;
     write(x,y);
     …
    end test;
```

```
    procedure B;
         real x, y; integer i, j;
    begin
     i:=5; y:=7.0;
     call test(i, j);
     …
    end B;
begin
    i:=4; y:=12.5; x:=0.0;
    call B;
    write (x);
end A;
```

Uppgift 5. (4p)

a) Vad menas med en *variabel på högen* (heap variable)?

b) Förklara varför sådana är nödvändiga i så många språk.. Vilken funktion fyller de som ”vanliga” variabler ej kan klara av? Jämför kortfattat ML och Pascal med avseende på förekomst och användande av heapvariabler.

c) Vad är en ”*dangling reference*”? Ge exempel på hur sådana kan uppkomma. Varför kan sådana vara farliga?

Uppgift 6. (3p)

Givet förljande program

```
PROGRAM Voo;
    VAR
    i : INTEGER;
    v : ARRAY [1..2] OF INTEGER;
    PROCEDURE Doo (x,y : INTEGER);
    BEGIN
     y := x;
     x := y + 2;
     i := i - 1;
     x := x+y;
    END

BEGIN
    i := 1;
    v[1] := 2;
    v[2] := 3;
    Doo(v[i], i);
    WriteLn (i, v[1], v[2]);
END.
```

Tala om vad som skrivs ut och varför om parametrarna är

a) value-parametrar

b) variable/referens-parametrar

c) value-result-parametrar

Uppgift 7. (3p)
Vi har talat om tre olika varianter av polymorfism under kursen:

- ad hoc-polymorfism
- inclusion polumorfism
- parametriserad polymorfism

Beskriv dessa tre former och ge exempel på hur de fungerar.

Uppgift 8. (3p)
a) Vad är ett *undantag* (an exception) i programspråkssammanhang? Vad gör man när ett undantag uppträder?
b) Vilka mål vill man uppnå med att införa undantag i ett programspråk?

Uppgift 9. (3p)
Ge tre olika kriterier som man bör ta hänsyn till när man väljer ett programspråk för ett större programvaruprojekt. Förklara vad de innebär.

Uppgift 10. (4p)
a) Förklara vad som menas med ett språks syntax och semantik. Förklara även skillnaden mellan abstrakt och konkret syntax.
b) Ange några fördelar med att använda en formell semantisk metod vid beskrivning av ett språk.

Uppgift 11. (4p)
Vi har diskuterat fyra olika *semantiska principer* som en språkdesigner bör ha i åtanke för att få ett regelbundet språk. Redogör för två av dessa och hur dessa påverkar språkdesign. (Det är inte namnen som är det viktiga utan principerna som sådana.)

Uppgift 12. (3p)
Är objektorienterade språk bra som nybörjarspråk? Motivera! (Det är motiveringen som bedöms.)