KTH, NADA, Henrik Eriksson

2D1320, Tilda, Tentamenslösning 12 april 1997

1. *Abstrakt stackstorlek*

Rekursiv tanke: För att få reda på antalet poster i en stack poppar man ett element från stacken och sparar i en lokal variabel. Svaret är 1 + antalet poster i den återstående stacken, men innan man returnerar värdet pushar man tillbaka det poppade elementet. . . .men om stacken är tom är svaret noll!

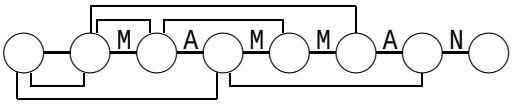
2. *Molekylvikten*

Rekursiv tanke: *Avslöjas inte – ingår i labb sju!*

Låt a=1, b=2,. . ., z=26 och A=0, B=27, C=54,. . ., Z=675. Då blir `Hash("A")=0` och `Hash("Zz")=701` – det behövs alltså en vektorlängd på 702.

Trädet i figuren har bara två löv.

3. *Leta efter mamman!*



i	next[i]
1	0
2	1
3	0
4	2
5	1
6	3

En snabbare metod är att titta på var sjätte bokstav i Strindbergs samlade verk. Så länge den inte är M, A eller N kan man hoppa vidare. När man träffar på ett M, A eller N får man kolla några bokstäver bakåt. Sedan hoppar man vidare från ett nytt utgångsläge.

4. *SL trafiksvar 08-600 06 00*

Bästaförstsökning med prioritetskö är bäst. Stamfar är en post med tre fält: time (13.00), stop (9141), och father (NIL). Söner skapas med en resa utan byten. För att slippa dumsöner krävs en array `best` där `best[9876]=14.17` anger att vi redan sett hur man kommer dit 14.17 och bara är intresserade av lägre tider.

Klockslagen lagras bäst som `CARDINAL`, nämligen antalet minuter efter kl 03.00.

Trappmodul och tidtabellsmodul är lämpligt att ha. Huvudprogrammet har förstås en procedur `MakeSons` och en `WriteChain` som anropas så snart den post som hämtas från trappan är resmålet.

5. *Lönar sej sortering?*

Osorterad vektor kan kräva en miljon jämförelser för varje sökning, dvs en miljard totalt. Med normal otur tar det hälften så mycket, alltså femhundra miljoner. Sortering med quicksort kräver $2N \ln N$ jämförelser, dvs cirka trettio miljoner. Sedan tar varje binärsökning bara tjugo jämförelser, dvs tjugotusen totalt. Det lönar sej att sortera!

6. *Syntax för nummeruppräkningar*

```
<uppräkning> ::= Nr <tal>. | Nr <tal> <svans>
<svans>      ::= och nr <tal>. | , nr <tal> <svans>
<tal>        ::= 1 | 2 | 3 |...
```

7. *Abstrakta betyg*

`TEXT` är inte bra om man vill räkna betygsmedel. `ARRAY OF INTEGER` är inte bra om man vill veta datum och studentdata.

En abstrakt `Betyg.T` kan vara en objekttyp med till exempel följande gränssnitt.

```
INTERFACE Betyg
TYPE T<:AbstraktBetyg;
     AbstraktBetyg = OBJECT
     METHODS
       registrera(kurs: Kurs.T; datum: Datum.T);
       isregistrerad(kurs: Kurs.T):BOOLEAN;
       putbetyg(kurs: Kurs.T; datum: Datum.T; betyg:INTEGER);
       getbetyg(kurs: Kurs.T):INTEGER;
       getdatum(kurs: Kurs.T):Datum.T;
       getstudent(): Student.T;
     END;
END Betyg.
```