KTH, NADA, Henrik Eriksson

2D1320, Tillämpad datalogi, Tentamenslösning 31 aug 2000

1. *Värstingvrålsautomat*

i	next[i]
1	0
2	1
3	0
4	2
5	1
6	0
7	1
8	0
9	2
10	5

2. *Värre än värst*

För tomt träd görs ingenting. Om rotordet är mindre än värst sätts roten till det rensade högerträdet, alltså `root:=root.right; Worse(root,"värst");`. Annars rensas vänsterträdet med `Worse(root.left,"värst")`.

3. *Värsta stacken*

Poppa stackarna omväxlande och putta in i kön. När ena stacken blir tom pushas kön på andra stacken.

4. *Värsta sökningen*

För lyckad sökning respektive misslyckad sökning kräver i genomsnitt

1. en ordnad vektor 10 resp. 11 jämförelser,
2. en kö 1000 resp 2000 jämförelser,
3. ett binärträd minst 10 resp 11 jämförelser, mer om det är obalanserat,
4. en hashtabell drygt 1 jämförelse i båda fallen.

5. *Värsta vännen*

Du är stamfar i ett problemträd där sönerna är dina vänner etc. Breddenförstsökning skapar hela trädet med en kö. Dumsonträd ser till att trädet blir ändligt. Sista posten i kön är din värsta vän.

6. *Värsta webbsyntaxen*

```
<webbfil>     ::= <text> | <Q> <webbfil> </Q> | <webbfil><BR><webbfil>
<Q>           ::= "<Q>"
</Q>          ::= "</Q>"
<BR>          ::= "<BR>"
```

7. *Verstingen*

Alla poeter in i en trappa, 16 ut ur trappan, in i en kö och åter till trappan, 8 ut ur trappan etc.

8. *Inte verst abstrakt*

Med abstrakta typen `Distance` är det inte så stor risk att det blir hopblandning av olika enheter.

Om `Distance` är en objekttyp kan den ha metodanropen

```
setverst(REAL x);
setkm(REAL x);
getverst():REAL;
getkm():REAL;
add(Distance d1);
greaterthan(Distance d1):BOOLEAN;
```