1. [**2p**]

Analysera tidskomplexiteten för följande program. Ge en exakt lösning. Antag att F() och write() tar konstant tid.

```
procedure something(n: integer);
var i,j,k : integer;
begin
    for i := 1 to n do begin
        for j := 1 to i do begin
            if F(j) then write(j)
        end
    end
end
```

2. [**2 + 2 = 4p**]

(a) Definiera ADT *Queue* (modell + operationerna - förklara vad operationerna gör).

(b) Beskriv implementationen av ADT *Queue* med circulär array. Beskriv datastrukturen. Visa hur en tom kö ser ut. Visa hur en full kö ser ut. Ange tidskomplexitet för varje operation.

3. [**2 + 3 = 5p**]

(a) Vilket resultat ger en traversering av trädet i figur 1 om traverseringen görs
(i) preorder?
(ii) inorder?
(iii) postorder?
(iv) levelorder (bredden-först)?

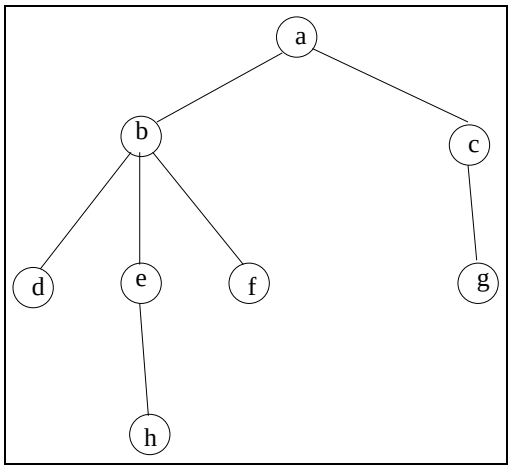

Figure 1: traverse

(b) Skriv en algoritm i pseudokod som använder ADT *Stack* för att utvärdera ett binärt uttryck i postfix format. Visa hur algoritmen fungerar genom att visa hur stacken ändras när man exekverar algoritmen för följande uttryck:

1 2 3 $*$ + 1 3 + $*$

4. [**2 + 1 = 3p**]

(a) Givet hashtabellen i figur 2 och nedanstående operationer. Antag stängd hashing och linjär sondering som kollisionshanteringsstrategi. Visa hur tabellen ser ut efter VARJE operation. (Operationerna körs i den givna ordningen.) Använd de givna hashvärdena h(x).

0	*empty*
1	*empty*
2	'I'
3	'DREAM'
4	'ABOUT'
5	'DALG'
6	*empty*

Figure 2: hashing

(i)	$Insert(A,$'LECTURES')	h('LECTURES') = 3
(ii)	$Delete(A,$'ABOUT')	h('ABOUT') = 2
(iii)	$Insert(A,$'DURING')	h('DURING') = 3

(b) Förklara dubbelhashing.

5. [**2p**]

Antag 2-3-trädet i figur 3.

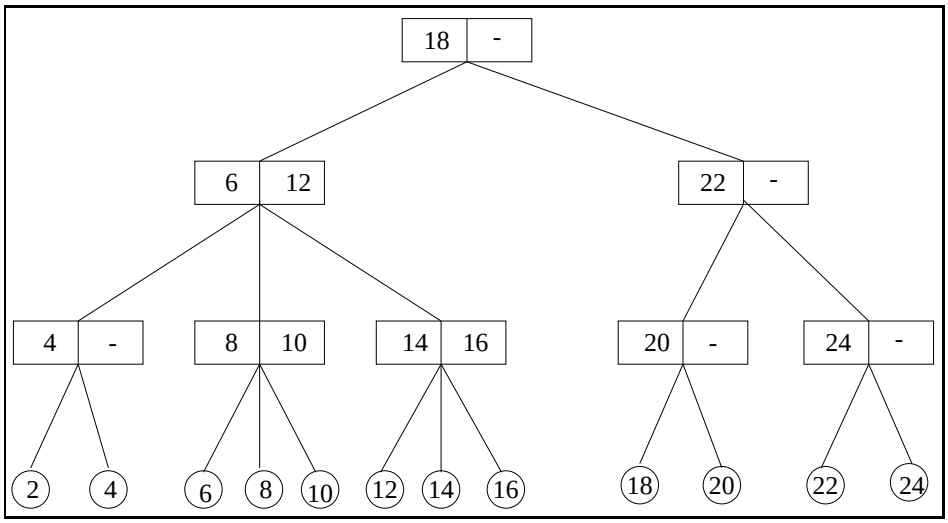

Figure 3: 2-3-tree

(i) Visa hur trädet ser ut efter exekvering av $Insert(15, A)$.

(ii) Visa hur trädet ser ut efter exekvering av $Delete(18, A)$ (utan $Insert$ i fråga (i)).

6. [$\mathbf{1 + 1 = 2p}$]

(a) Vad är det maximala antalet noder för
(i) ett perfekt binärt träd med höjd 3
(ii) ett vänsterfullständigt binärt träd med höjd 3, som inte är perfekt

(b) Rita Fibonacci-trädet T_2.

7. [**3 + 3 + 1 = 7p**]

(a) Visa hur *QuickSort* (med initial swap) sorterar sekvensen

$$< \text{I,L,O,V,E,D,A,L,G} >$$

Låt det största av de första två skillda nycklar vara *pivot*.

(b) Visa hur sekvensen sorteras med hjälp av *HeapSort*.

(c) Ange tidskomplexitet för *QuickSort* och *HeapSort* i värsta fall OCH medelfallet.

8. [**2p**]

Lös följande rekursiv ekvation.

$$\begin{array}{lcll} T(n) & = & d & \text{if } n = 1,2 \\ & & T(n/3) + c & \text{if } n \geq 3 \end{array}$$