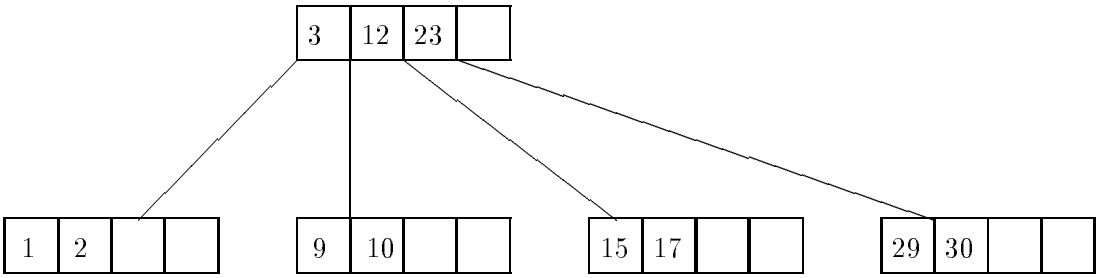
Løsningsforslag og sensorveiledning til eksamen i IN110 våren 1992

Generelt:

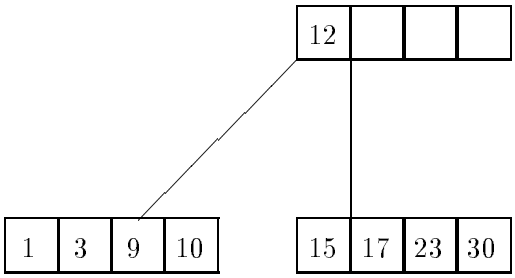
Oppgaven består av tre hoveddeler, som er vektet med hhv. 10, 45 og 45 prosent. Innenfor hver hoveddel skal deloppgavene vektes likt.

Besvarelsene kan være i enten SIMULA eller PASCAL. Det er allikevel den algoritmiske forståelsen vi er ute etter, og det skal derfor ikke legges vekt på syntaksfeil, og f.eks forsøk på å benytte en REPEAT konstruksjon i SIMULA.

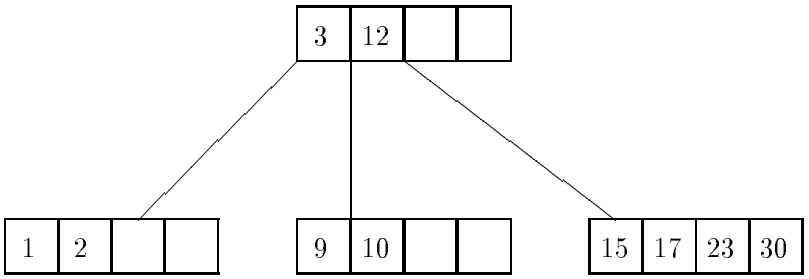
1-a



1-b



For de som har oversett at 29 skal fjernes *i stedet* for å sette inn 2, vil løsningen se slik ut:



Denne løsningen skal gis karakteren 1.5.

2-a

```
ref(flytt) PROCEDURE finnflytt(brett);
CHARACTER ARRAY brett;
BEGIN
  INTEGER i;
  i := 1;
  WHILE brett(i) <> 'L' DO i := i+1;

  IF i-1 >0 AND THEN brett(i-1) = 'H' THEN BEGIN
    peker :- new flytt;
    peker.fra := i-1;
    peker.til := i;
    (peker.neste := NONE);
  END;

  IF i-2 > 0 AND THEN brett(i-2) = 'H' THEN BEGIN
    hjelper :- new flytt;
    hjelper.neste := peker;
    peker :- hjelper;
    peker.fra := i-2;
    peker.til := i;
  END;

  IF i+1 <= lengde AND THEN brett(i+1) = 'S' THEN BEGIN
    hjelper :- new flytt;
    hjelper.neste := peker;
    peker :- hjelper;
    peker.fra := i+1;
    peker.til := i;
  END;

  IF i+2 <= lengde AND THEN brett(i+2) = 'S' THEN BEGIN
    hjelper :- new flytt;
    hjelper.neste := peker;
    peker :- hjelper;
    peker.fra := i+2;
    peker.til := i;
  END;

  finnflytt :- peker;
END;
```

2-b

```
PROCEDURE generer(i); INTEGER i;
BEGIN
  REF(flytt) liste;
  IF i = 0 THEN skrivut(brett)
  ELSE BEGIN

     liste :- finnflytt(brett);

     WHILE liste =/= NONE DO BEGIN
       bytt(liste.fra, liste.til);
       generer(i-1);
       bytt(liste.fra, liste.til);
       liste :- liste.neste;
     END;

  END;
END;
```

2-c

Dette er den optimale løsningen med bredde først søk simulert ved gjhentagne dybde først søk:

```
PROCEDURE finnspill(situasjon,k);
CHARACTER ARRAY situasjon;
INTEGER k;
BEGIN
  BOOLEAN funnet;
  INTEGER ARRAY flyttsekvens(1:k);

  PROCEDURE generer(i, situasjon); INTEGER i;
                                   CHARACTER ARRAY situasjon;
  BEGIN
    REF(flytt) liste;
    IF i = 0 THEN BEGIN
       IF identiske(brett, situasjon) THEN funnet := TRUE;
    END
    ELSE IF NOT funnet THEN BEGIN

       liste :- finnflytt(brett);
       WHILE liste =/= NONE AND NOT funnet DO BEGIN
         bytt(liste.fra, liste.til);

         flyttsekvens(i) := liste.fra;

         generer(i-1);
         bytt(liste.fra, liste.til);
         liste :- liste.neste;
       END;

    END;
  END;

  INTEGER j;

  funnet := FALSE; j := 0;
  WHILE NOT funnet and j <= k DO
  BEGIN
    generer(j, situasjon);
    j := j+1;
  END;

  IF funet THEN  FOR j := j-1 STEP -1 UNTIL 1 DO
  outint(flyttsekvens(j),3);
END;
```

Alternativ løsning på 2-c. Denne løsningen garanterer ikke at du finner det korteste spillet først. Den skal gi karakteren 1.5:

```
PROCEDURE finnspill(situasjon,k);
CHARACTER ARRAY situasjon; INTEGER k;
BEGIN
  INTEGER kortest, j;
  INTEGER ARRAY flyttsekvens(1:k), hittilkortest(1:k);

  PROCEDURE generer(i, situasjon); INTEGER i;
                                   CHARACTER ARRAY situasjon;
  BEGIN
    REF(flytt) liste;
    IF i = 0 THEN BEGIN
       IF identiske(brett, situasjon) THEN  BEGIN
          kortest := k-i;
          FOR j := 1 STEP 1 UNTIL kortest DO
            hittilkortest(j) := flyttsekvens(j);
       END;
    END

    ELSE IF k-i+1 < kortest THEN BEGIN   ! Har ikke et kortere spill fra for;
       liste :- finnflytt(brett);

       WHILE liste =/= NONE  BEGIN
         bytt(liste.fra, liste.til);

         flyttsekvens(i) := liste.fra;

         generer(i-1);
         bytt(liste.fra, liste.til);
         liste :- liste.neste;
       END;

    END;
  END;

  kortest := k+1;
  generer(k, situasjon);
  IF kortest <= k THEN
     FOR j := j-1 STEP -1 UNTIL 1 DO outint(hittilkortest(j),3);

END;
```

Dersom man i denne løsningen kun har avskjæring ved dybde `k`, og ikke ved den til nå korteste veien skal det gies karakteren 2.5.

2-d I bredde først med dybde først varianten kan følgende svares:

Den modifiserte varianten av prosedyren generer kunne som postfiks-operasjon skrive ut det flyttet den nå var kommet til dersom funnet = TRUE. Dette ville gitt flyttene i baklengs rekkefølge.

I den andre varianten er dette ikke mulig fordi men ikke er garantert å treffe på den beste løsningen først. Det beste svaret for de som har programmert denne varianten i oppgave 2-c vil være om de her beskriver BFS ved DFS, og legger til det samme som beskrevet over. Det nest beste er at de svarer "NEI, det er ikke mulig fordi.....". Denne siste varianten skal verdsettes til 1.5.

De som har svaralternativ 2 i oppgave 2-c, og som likevel svarer at man kan skrive ut som postfiksoperasjon skal ha karakteren 4.0.

3-a

```
PROCEDURE generer(n);
REF(situasjon) n;
BEGIN
  REF(trekk) l;
  INTEGER j;

  l :- finnflytt(brett);
  j := 1;

  WHILE l =/= NONE DO
  BEGIN
    bytt(l.til,l.fra);
    n.nestesituasjon(j) :- finnode(brett);

    IF n.nestesituasjon(j) == NONE DO
    BEGIN
      n.nestesituasjon(j) :- NEW situasjon;
      n.nestesituasjon(j).brett := brett;   % eller noe lignende.
      generer(n.nestesituasjon(j));
    END

    j := j+1;

    bytt(l.til,l.fra);
    l :- l.neste;
  END;

END;
```

3-b

F. eks en søketre-struktur. Dette krever at klassedeklarasjonen utvides noe. Hashing er litt vanskelig, da vi ikke vet hvor mange noder det er snakk om.

3-c

```
BOOLEAN PROCEDURE strip(n);
REF(node) n;
BEGIN
  INTEGER j;
  BOOLEAN utkanter;

  IF n == NONE THEN strip := FALSE
  ELSE IF vinnersituasjon(n) THEN strip := TRUE
  ELSE  BEGIN

    FOR j := 1 STEP 1 UNTIL 4 DO
      IF NOT strip(n.nestesituasjon(j)) THEN
         n.nestesituasjon(j) := NONE
      ELSE utkanter := TRUE;

      strip := utkanter;

  END;

END;
```

3-d

```
PROCEDURE snu(n);
REF(node) n;
BEGIN
  INTEGER i,j;

  n.merket := TRUE;

  FOR i := 1 STEP 1 UNTIL 4 DO
  BEGIN
    IF NOT n.nestesituasjon.merket DO
      snu(n.nestesituasjon);
    j := 1;
    WHILE n.nestesituasjon(j) =/= NONE DO j := j+1;
    n.nestesituasjon(j) :- n;
  END;

  FOR I := 1 STEP 1 UNTIL 4
    n.nestesituasjon(i) :- NONE;
END;
```

Eventuelt noe mer elegant med en integer merket initialisert til 0 i alle noder:

```
PROCEDURE snu(n);
REF(node) n;
BEGIN
  INTEGER i;

  n.merket := n.merket + 1;

  % Dersom vi er her for f|rste gang, skal det kalles rekursivt.
  IF n.merket = 1 THEN
    FOR i := 1 STEP 1 UNTIL 4 DO
    BEGIN
      IF n.nestesituasjon(i) =/= NONE THEN snu(n.nestesituasjon(i));

      % Snu pekeren
      n.nestesituasjon(i).nestesituasjon(n.nestesituasjon(i).merket) :- n;
    END

  % Blank ut denne nodens pekere s} de kan benyttes av den
  % kallende prosedyreinstansen.
  FOR i := 1 STEP 1 UNTIL 4
    n.nestesituasjon(i) :- NONE;
END;
```