```
ÕÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¸
³ Tentamen Algoritmiek, 7 augustus 2001, 9.00-12.00 uur  ³
ÔÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¾
```

OPMERKING. Geef een duidelijke toelichting bij al je antwoorden!!!

Puntenverdeling: 1: 25; 2: 25; 3: 25; 4: 25;

1. We bekijken een variant op het Nim-spel. Op tafel liggen n (>1) lucifers. Er zijn twee spelers, Herman en Nina. De speler die begint mag minimaal 1 en maximaal n-1 lucifers van tafel nemen. Vervolgens moeten de spelers om de beurt minimaal 1 lucifer en maximaal twee keer het aantal lucifers dat de tegenstander zojuist heeft gepakt van tafel nemen. De speler die de laatste lucifer(s) wegneemt heeft gewonnen.
   Voorbeeld: stel dat Nina op een gegeven moment twee lucifers van tafel neemt. Dan mag Herman in de volgende zet 1, 2, 3 of 4 lucifers pakken, maar niet meer.

(a) Wat zijn voor dit spel toestanden en acties?

(b) (i) Wanneer is bij dit spel een toestand direct (dat wil zeggen in ,,n zet) gewonnen door de speler die aan de beurt is?
   (ii) Geef de toestand-actie-graaf voor het geval n=7. Neem aan dat Nina begint. Direct gewonnen toestanden (zie (i)) hoef je niet verder uit te werken, maar schrijf er wel bij wie wint. Is het spel voor n=7 winnend voor de beginnende speler? (d.w.z. kan Nina altijd winnen, ongeacht wat Herman doet?) Geef daartoe voor alle directe vervolgstanden van de beginstand aan of deze winnend of verliezend zijn voor de speler die aan de beurt is (Herman in dit geval).

In de onderdelen (c) en (d) bekijken we het spel weer voor algemene n.

(c) Geef een C++-datastructuur (klasse) toestand die alle informatie van een stand kan bevatten (voor het algemene geval, waarbij met (const) n lucifers gestart wordt).

(d) Schrijf een C++-functie void vervolg(toestand stand) (een memberfunctie stand.vervolg() mag ook) die alle standen afdrukt die vanuit een gegeven toestand stand in ,,n zet te bereiken zijn (algemene n). Test daarin eerst of stand direct gewonnen is. Zo ja, dan hoeven niet alle vervolgstanden te worden afgedrukt, maar dan moet wel gemeld worden dat direct gewonnen kan worden en wat de winnende zet in dat geval is. Neem aan dat een member-functie stand.drukaf(), die de toestand stand op het beeldscherm afdrukt, beschikbaar is.

2. In het holst van de nacht breekt inbreker en meesterkraker A.M. de kluis van een miljonair uit W. open. In de kluis treft hij n zeer waardevolle voorwerpen aan. Van elk voorwerp zijn de waarde en het gewicht bekend: item i is Ei euro waard en weegt Ki kilo (Ei, Ki geheel en > 0).
   De inbreker wil natuurlijk een zo waardevol mogelijke buit meenemen, maar helaas kan hij in zijn rugzak slechts voor een totaalgewicht aan hooguit T kilo meesjouwen.
   De vraag is nu: welke voorwerpen moet hij in zijn rugzak meenemen om een zo hoog mogelijke totaalwaarde te verkrijgen (onder bovengenoemde restrictie), en wat is dan die maximale waarde?

(a) Geef in woorden een algoritme dat een optimale buit (dus met maximale totaalwaarde) vindt. Het algoritme moet gebruik maken van de methode backtracking. Leg duidelijk uit waar de backtracking plaatsvindt.

Een andere oplossingsmethode is om top-down met gebruik van recursie tot een oplossing te komen. Voor het gemak nemen we vanaf hier aan dat we alleen op zoek zijn naar de maximale waarde van de buit onder de gegeven

restrictie, en niet in de buit zelf.

(b) Laat buit(i,j) de maximale totaalwaarde van de buit aangeven als alleen uit de voorwerpen 1 t/m i gekozen mag worden, en het totale gewicht van de gekozen voorwerpen <=j moet zijn (1 <= i <= n, 0 <= j <= T). Dan geeft buit(n, T) de voor de inbreker interessante maximale waarde van de buit. Leg uit waarom buit(i,j) aan de volgende recurrente betrekking voldoet.

```
buit(i,0) = 0     als 1 <= i <= n

buit(1,j) = 0     als 0 < j < K1
buit(1,j) = E1    als K1 <= j <= T

buit(i,j) = buit(i-1,j)              als 1 < i <= n en 0 < j < Ki
buit(i,j) = maximum ( buit(i-1,j), Ei + buit(i-1, j-Ki) )
                                     als  1 < i <= n en Ki <= j <= T
```

(c) Gebruik makend van de recurrente betrekking uit (b) kan buit(n,T) recursief berekend worden. Geef aan wat de basisgevallen van de recursie zijn, en welke recursieve aanroepen gedaan worden. Leg tenslotte duidelijk uit waarom deze methode niet efficient is.

De laatste oplossingsmethode maakt gebruik van dynamisch programmeren. Hierbij wordt een array gebruikt dat bottom up (van klein naar groot) wordt ingevuld. In dit geval is dat een tweedimensionaal array buit, waarbij buit[i][j] de maximale buit is die verkregen kan worden als je alleen uit de voorwerpen 1 t/m i kan kiezen en je in totaal voor hooguit j kilo mee kan nemen. De buit[i][j]'s voldoen dan uiteraard aan een recurrente betrekking als boven, met buit[i][j] i.p.v. buit(i,j). De gevraagde oplossing komt uiteindelijk in buit[n][T] te staan.

(d) Geef in pseudocode of in C++ een algoritme dat het array vult met de juiste waarden. Hoeveel stappen doet dit algoritme? Kun je ook met een kleiner hulparray toe als je alleen de waarde buit[n][T] wilt weten en later niet meer in de tussenresultaten geinteresseerd bent?

3. Gegeven een binaire boom, toegankelijk via de pointer wortel van type knoop*, die integers bevat:

```
struct knoop {
    int info;
    knoop* links;
    knoop* rechts;
};  // knoop
```

Veronderstel dat de info-velden allemaal verschillend zijn.

We definieren een stamboom als een binaire boom die voldoet aan de volgende twee eigenschappen:

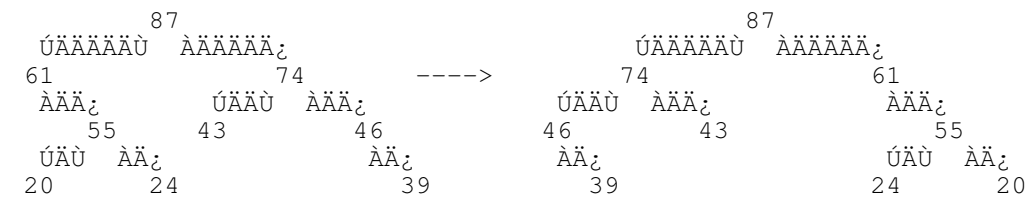
1. in elke knoop geldt de heap-eigenschap (d.w.z het info-veld van die knoop is groter dan dat van zijn eventuele kinderen)
2. voor elke knoop met twee kinderen geldt dat het info-veld van het linkerkind groter is dan dat van het rechterkind.

(a) Hoeveel stambomen die tevens compleet zijn bestaan er met daarin de sleutels 1 t/m 7? Motiveer je antwoord.

(b) Schrijf een RECURSIEVE C++-functie bool stamboom(knoop* wortel), die TRUE oplevert als een gegeven binaire boom een stamboom is en anders FALSE.

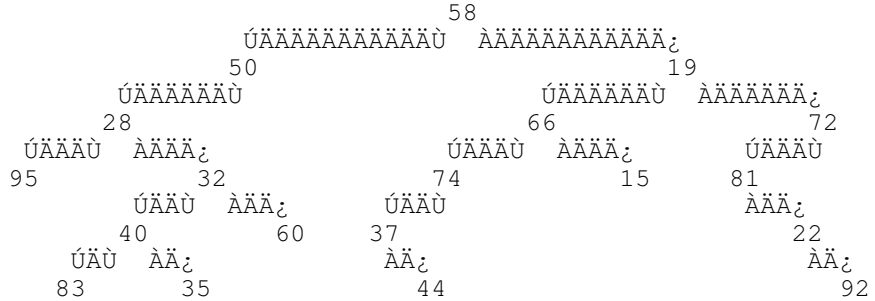
(c) Gegeven is een binaire boom die voldoet aan eigenschap 1. Van deze boom kan nu eenvoudig een stamboom worden gemaakt door in elke knoop de linker- en rechtersubboom te verwisselen indien het info-veld van het linkerkind kleiner is dan dat van het rechterkind.
Schrijf een RECURSIEVE C++-functie void maakstamboom(knoop* wortel), die op deze manier van een gegeven binaire boom (die aan eigenschap 1 voldoet) een stamboom maakt.

Voorbeeld: De functie maakstamboom losgelaten op de binaire boom hier linksonder levert de binaire boom rechtsonder.

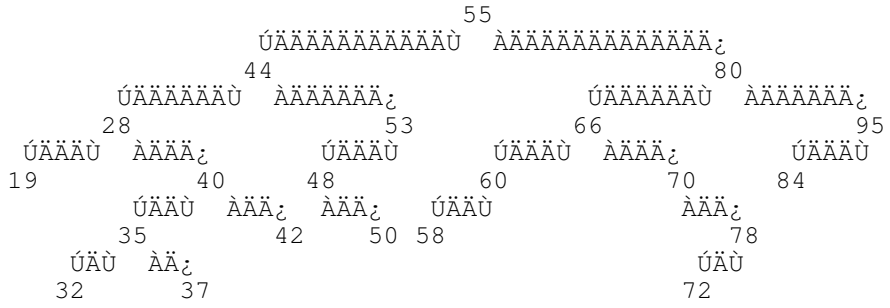


4. Deze opgave gaat over binaire bomen. De verschillende onderdelen staan echter los van elkaar.

(a) Zoals bekend bestaat er een 1-1-correspondentie tussen geordende bossen en binaire bomen. Geef nu het geordende bos dat correspondeert met onderstaande binaire boom.



(b) Geef de inhoud van de knopen van bovenstaande binaire boom in symmetrische (LWR) volgorde.
(c) Hoe ziet de binaire ZOEKboom eruit die precies dezelfde vorm heeft als de boom in (a), en bovendien dezelfde sleutels bevat (te weten: 15, 19, 22, 28, 32, 35, 37, 40, 44, 50, 58, 60, 66, 72, 74, 81, 83, 92, 95)?
(d) Welke binaire zoekboom ontstaat als we aan onderstaande binaire zoekboom achtereenvolgens 75 en 25 toevoegen? Toevoegen gaat via het gebruikelijke toevoegalgoritme voor binaire zoekbomen.



(e) Welke binaire zoekboom ontstaat als we uit de in (d) ontstane binaire zoekboom achtereenvolgens 53 en 80 verwijderen? Verwijderen gaat via verwisselen met de grootste kleinere indien nodig.

VEEL SUCCES !

□(8U