(10U -1-

-1-

```
ÕÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¸
³ Tentamen Algoritmiek, 5 juni 2000, 9.00-12.00 uur ³
ÔÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¾
```

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

Puntenverdeling:
1: 35 2: 15 3: 30 4: 20

1. We bekijken een variant op het Nim-spel. De spelregels zijn als volgt. Op tafel liggen m (>=1, geheel) stapels lucifers. Er zijn twee spelers, Romeo en Julia. De spelers mogen om de beurt van ,,n van de stapels 1 of 2 lucifers pakken. Het spel is afgelopen als alle stapels leeg zijn. De speler die de laatste lucifer(s) wegneemt heeft gewonnen. We spreken af dat Romeo begint.

(a) Wat zijn voor dit spel toestanden en acties (voor algemene waarden van m en de aantallen lucifers op de stapels)?

(b) Teken het toestand-actie-diagram voor het geval dat m = 2 en er drie lucifers op elk van beide stapels liggen.
Is het spel in dit geval (dus m=2 en 3 lucifers op elke stapel) winnend voor Romeo of voor Julia? Hoe moet hij/zij dan spelen om te winnen?
Opm. 1. Je hoeft in je toestand-actie-diagram toestanden die meer dan ,,n keer voorkomen maar ,,n keer uit te werken.
Opm. 2. Een situatie met (bijvoorbeeld) 3 lucifers op de eerste stapel en 2 op de tweede is natuurlijk geheel symmetrisch aan het geval dat er 2 op de eerste en 3 op de tweede stapel liggen. Je hoeft maar ,,n van die twee gevallen uit te werken.

(c) Iets algemener: is het spel voor m = 2 en op beide stapels evenveel lucifers winnend voor Romeo of voor Julia, en wat is dan de winnende strategie?

(d) Geef een geschikte C++-datastructuur (klasse) toestand die alle informatie van een spelsituatie kan bevatten (voor algemene (const) m en aantallen lucifers op de stapels).

(e) Schrijf nu een C++-functie void volgende(toestand stand) die de directe vervolgtoestanden van een gegeven situatie stand afdrukt (weer voor algemene waarden van m en de aantallen per stapel).
Neem aan dat een member-functie stand.drukaf(), die de situatie stand op het beeldscherm afdrukt, beschikbaar is.

(f) Beschrijf kort hoe de onder (e) geschreven functie kan worden aangepast opdat deze recursief bepaalt of de gegeven situatie stand winnend is of verliezend voor de speler die aan de beurt is. Onderscheid basisgeval en recursieve (eenvoudiger) aanroepen.
Ga er hierbij vanuit dat een member-functie stand.eind(), die True is als stand een eindstand is (d.w.z. op alle stapels nul lucifers) en anders False, beschikbaar is.

.

-2-

2. Langs een rivier zijn n handelsposten gevestigd, stroomafwaarts genummerd 1, 2, 3, ..., n. Bij elke post kun je een kano huren, die bij een post verder stroomafwaarts weer kan worden teruggegeven. Voor elk mogelijk vertrekpunt i en elk mogelijk aankomstpunt j > i zijn de kosten van de huur van een kano bekend en opgeslagen in een array kano[i][j]. Het is mogelijk dat het goedkoper is om onderweg van i naar j op een of meer plaatsen de ene kano in te leveren en verder te gaan in een andere kano. Met andere woorden, het kan goedkoper zijn om met verschillende kano's van i naar j te gaan dan in een keer met de kano die in plek i gehuurd is. De bedoeling is natuurlijk om zo goedkoop mogelijk van 1 naar n te gaan. Gegeven is dat je alleen stroomafwaarts kunt varen.

(a) Geef (in woorden) een voor de hand liggend gretig algoritme voor het bepalen van de goedkoopst mogelijke reis. Waarom denk je dat je algoritme niet altijd leidt tot een optimale (= goedkoopste) oplossing?
Geef hierbij ter illustratie ook een klein voorbeeld.

(b) We willen nu de kosten van de goedkoopste kano-reis van 1 naar n berekenen via dynamisch programmeren. Zij hiertoe reis[i] (1<=i<=n) een array dat bottom up wordt ingevuld. Hierbij geeft reis[i] steeds de kosten van de goedkoopste kano-reis aan van post 1 naar post i. Het eindantwoord staat dan in reis[n]. Leg uit dat geldt:

```
reis[1] =    0

reis[i] =   minimum  { reis[k] + kano[k][i] }   (1 < i <= n)
            1<=k<i
```

(c) Geef in pseudocode of in C++ een algoritme dat het array reis bottom up vult. Leg daartoe eerst uit hoe reis[i] uit reeds eerder ingevulde array-elementen kan worden berekend. Gebruik hierbij de recurrente betrekking uit (b).

.

-3-

3. Gegeven een niet lege binaire boom, toegankelijk via de pointer wortel van type knoop*, die verschillende gehele getallen bevat:

```
struct knoop {
    int info;
    knoop* links;
    knoop* rechts;
    int reus;
```

```
        };  // knoop
```

De reus-velden hebben bij aanvang van deze opgave nog geen waarde.

(a) Schrijf een RECURSIEVE C++-functie void init(knoop* wortel) die van alle knopen het reus-veld vult met de info-waarde uit die knoop. (M.a.w. Copieer de info-velden in de reus-velden.)

(b) Laten de reus-velden geinitialiseerd zijn zoals in (a) beschreven. Schrijf nu een RECURSIEVE C++-functie void maxi(knoop* wortel) die in elke knoop de reus-velden vult met het maximum van de info-waarden uit de (sub)boom met die knoop als wortel.

Neem vanaf hier aan dat de reus-velden gevuld zijn zoals in (b) bedoeld.

(c) Schrijf een niet-recursieve C++-functie knoop* grootste(knoop* wortel), die een pointer oplevert naar de knoop die de grootste waarde uit de boom bevat. Gebruik hierbij de reus-velden.

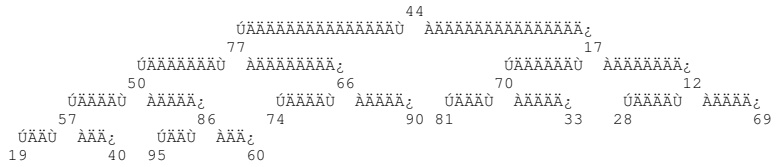
(d) Neem aan dat de boom ten minste twee knopen heeft. Schrijf nu een niet-recursieve functie knoop* op1na(knoop* wortel), die een pointer oplevert naar de knoop die de op ,,n na grootste waarde uit de boom bevat. Veronderstel hierbij dat de knoop waarin de op ,,n na grootste zich bevindt een afstammeling is van de knoop met de grootste waarde. Gebruik weer de reus-velden, en de functie uit (c).

.

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

(a) (i) Waar bevindt zich de grootste waarde in een heap (= hoopstructuur), en waar de op ,,n na grootste?
(ii) Laat zien dat er acht verschillende heaps bestaan met daarin de sleutels 1 t/m 5.

Gegeven is volgende complete binaire boom:



(b) Breng bovenstaande boom in hoopstructuur door toepassing van de methode heapify. Laat duidelijk de achtereenvolgende stappen zien.

(c) Bereken voor de voorbeeldboom de externe padlengte.

(d) Geef de binaire ZOEKboom die resulteert als de volgende getallen in de aangegeven volgorde aan een aanvankelijk lege zoekboom worden toegevoegd.

55, 71, 17, 28, 66, 60, 19, 74, 33, 49, 68, 90, 40, 88, 44, 12, 35, 77

Toevoegen gaat via het gebruikelijke toevoegalgoritme voor binaire zoek-bomen.

(e) Welke binaire zoekboom ontstaat als we uit de in (d) ontstane binaire zoekboom achtereenvolgens de waarden 74 en 55 verwijderen? Verwijderen gaat via het gebruikelijke verwijderalgoritme voor binaire zoekbomen, dus via verwisselen met de grootste kleinere indien nodig.

Veel succes!

(8U