Tentamen Programmeermethoden
Woensdag 28 maart 2012, 14.00–17.00 uur
Universiteit Leiden — Informatica

Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of als locale variabele voorkomen; vul zelf headings goed in. De opgaven tellen alle vier even zwaar mee. Succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/cijf/res.html`.

1. In een array `int A[n]` staan `n` (een `const` > 0) verschillende gehele getallen.
a. Schrijf een C++-functie `double gem (A,n)` die het (niet-afgeronde) gemiddelde van de getallen uit `A` teruggeeft.
NB Geef hier (en ook in de andere opgaven) de compleet ingevulde heading van de functie!
b. Schrijf een Booleaanse C++-functie `gesorteerd (A,n)` die precies dan `true` oplevert als `A` oplopend gesorteerd is.
c. Schrijf een C++-functie `wissel (A,i,j)` die als $0 \leq$ `i` $<$ `j` $<$ `n` en `A`[`i`] $>$ `A`[`j`], of als $0 \leq$ `j` $<$ `i` $<$ `n` en `A`[`i`] $<$ `A`[`j`], deze array-elementen verwisselt, en anders niets doet.
d. Schrijf een C++-functie `sorteer (A,k,n)` die het array `A` als volgt sorteert. Vergelijk (en verwissel zonodig) herhaald, met de functie van **c**, willekeurige elementen `A[i]` en `A[j]`. Controleer direct aan het begin, en na elke `k` (een geheel getal > 0) aanroepen van **c**, met de functie van **b** of `A` al gesorteerd is (en zo ja, stop).
Gebruik een functie `int ran ( )` die een random getal tussen `0` en `INT_MAX`, grenzen inbegrepen, oplevert (deze hoef je niet zelf te schrijven) om `i` en `j` te genereren.
e. Hoeveel vergelijkingen tussen array-elementen doet `sorteer` minimaal bij een omgekeerd gesorteerd rijtje, dit afhankelijk van de waarde van `k`?

2.a. Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *locale* en *globale* variabelen. Verder onderscheiden we ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.
b. Gegeven een C++-programma met daarin de volgende twee functies:

```
bool eva (int m, int n) {
  if ( m < n ) n--; else m--;
  x++; cout << x << ", " << m << " en " << n << endl;
  return ( m < n );
}//eva
int floris (int x, int y) {
  if ( eva (x,y) ) { x = x - y; y = x + y; x = y - x; }
  else { x = x + 10; y = y - 2; }
  cout << x << " en " << y << endl; return x+y;
}//floris
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `floris` wordt aangeroepen hebben zij de waarde 20, 3 en 4 respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = floris (y,z); cout << x << ", " << y << " en " << z << endl;
```

c. Als **b**, maar nu staat er een `&` bij alle vier parameters, en `x`, `y` en `z` starten op 20, 3 en 8.
d. Als **c** (dus met vier `&`'s), en `x`, `y` en `z` beginnen op 998, 3 en 8, voor

```
x = floris (x,x); cout << x << ", " << y << " en " << z << endl;
```

e. Je wilt nu binnen de functie `eva` de functie `floris` aanroepen. Wat moet je dan nog toevoegen in je programma en is er dan sprake van (indirecte) recursie?

3. Gegeven is een `n` bij `n` (een `const` > 1) array `K`, met gehele getallen. Hierbij geeft `K[i][j]` de kosten aan van een directe reis van `i` naar `j`, die even hoog zijn als die van `j` naar `i`; de directe reis van `i` naar zichzelf is gratis. Rechts staat een voorbeeld met `n` $= 4$.

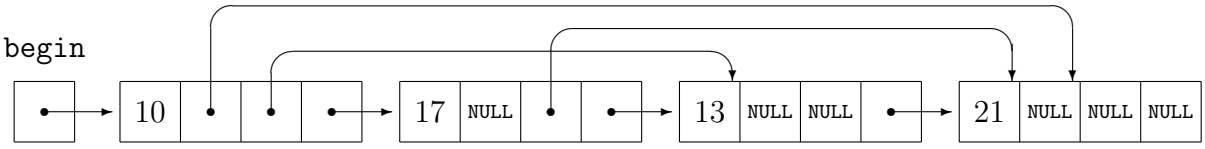
0	3	4	2
3	0	7	8
4	7	0	1
2	8	1	0

a. Schrijf een Booleaanse C++-functie `driehoek (K)` die precies dan `true` terugeeft als `K` aan de *driehoeksongelijkheid* voldoet: voor elk drietal plaatsen `i`, `j` en `k` geldt dat de directe reis van `i` naar `j` hooguit evenveel kost als de totale kosten van de route via `k`.
b. Alle kosten worden 20% lager. Schrijf een C++-functie `lager (K)` die dit tot stand brengt. Rond hierbij op de gebruikelijke wijze af: 8 wordt 6 (via 6.4), 7 wordt ook 6 (via 5.6).
c. Iemand besluit op de volgende ietwat wonderlijke manier te reizen: hij begint in plek `i` en maakt daarna herhaald de goedkoopste reis naar een andere plaats (in geval van meerdere mogelijkheden: de plaats met het laagste nummer). Hij stopt als hij een eerder bezochte plek bereikt. Schrijf een C++-functie `int totprijs (K,i)` die de totale kosten uitrekent van deze reis.
Tip: gebruik een Booleaans array om bij te houden waar de reiziger al geweest is. In het voorbeeld, met `i` $=$ `0`, is het $2 + 1 + 1 = 4$ wegens de reis van 0 naar 3 naar 2 naar 3.

4. Gegeven is het volgende type:

```
class pr { public: int prijs; pr* volg1; pr* volg2; pr* volg3 };
```

Met behulp hiervan worden lijstjes met prijzen opgebouwd. Het veld `volg1` bevat een pointer naar het volgende `pr`-object, `volg2` wijst naar het daarop volgende object, en `volg3` naar het daarop volgende (soms `NULL`). Een voorbeeld (`begin` van type `pr*`; in de objecten staan de pointers in volgorde `volg3`, `volg2`, `volg1` getekend):



a. Schrijf een C++-functie `verwijder (begin)` die het eerste `pr`-object uit de lijst (met `begin` van type `pr*` als ingang) netjes verwijdert, mits dat object bestaat en de prijs erin even is.
b. Schrijf een C++-functie `voegtoe (begin,prijsje)` die een nieuw `pr`-object met prijs `prijsje` erin vooraan de lijst met ingang `begin` toevoegt. Zet wederom alle pointers goed.
c. Schrijf een C++-functie `verwissel (begin)` die de eerste twee `pr`-objecten uit de lijst met ingang `begin` verwisselt, mits deze objecten bestaan. Let op: verwissel de objecten, niet de inhouden! Zet uiteraard wel alle pointers goed.
d. In de functies bij **a**, **b** en **c** staat in de heading een pointer. Deze heb je call by value of call by reference doorgegeven (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Leg duidelijk uit. Zou het uitmaken als bij **c** alleen de twee prijs-inhouden verwisseld moesten worden?
e. Schrijf een C++-functie `pr* laatste (begin)` die een pointer naar het laatste `pr`-object uit de lijst met ingang `begin` oplevert. Als de lijst leeg is, moet `NULL` worden teruggegeven. Er moeten zo weinig mogelijk pointers afgelopen worden.