# Tentamen Programmeermethoden
# Woensdag 4 januari 2012, 10.00–13.00 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 Booleaanse C++-functie `gesorteerd (A,n)` die precies dan `true` oplevert als `A` oplopend gesorteerd is.
**b.** Schrijf een C++-functie `draai (A,i,j)` die het array-gedeelte `A[i]`,...,`A[j]` omdraait. Zo moeten onder andere `i`-de en `j`-de array-element verwisseld worden. Neem aan dat $0 \leq$ `i` $<$ `j` $<$ `n`. Dus 1 6 4 5 3 8 wordt 1 3 5 4 6 8 (als `i` $= 1$ en `j` $= 4$).
**c.** Voor een gegeven array `A` noemen we het minimale aantal aanroepen van de functie van **b** dat nodig is om het array oplopend te sorteren de *verwarringsgraad* van `A`.
Schrijf een Booleaanse C++-functie `een (A,n)` die precies dan `true` oplevert als `A` een verwarringsgraad van 1 heeft. Gebruik alleen de functies van **a** en **b** om `A` te benaderen. Hint: probeer "alle" (`i`, `j`)-combinaties.
**d.** Schrijf een C++-functie `sorteer (A,n)` die het array `A` oplopend sorteert met behulp van *bubblesort*. Hierbij moet het algoritme stoppen zodra tijdens een ronde (doorgang) geen verwisselingen hebben plaats gehad. Gebruik de functie van **b**.
**e.** Hoeveel vergelijkingen tussen array-elementen doet `sorteer` minimaal? En maximaal?

**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:

```
int kimbra (bool jn, int u, int y) {
  if ( ! jn ) { cout << "Som" << endl; u--; return (u+1) + y; }//if
  else { cout << "Product" << endl; u -= 2; return (u+2) * y; }//else
}//kimbra
int gotye (int a, int b) {
  int i; bool jn;
  for ( i = 1; i <= a; i++ ) {
    jn = ( i < b ); b += kimbra (jn,a,b); cout << a << "," << b << endl; }//for
  return a % b;
}//gotye
```

Verder zijn de globale variabelen `u`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `gotye` wordt aangeroepen hebben zij de waarde 7, 3 en 0, respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
u = gotye (y,z); cout << u << "," << y << "," << z << endl;
```

**c.** Als **b**, maar nu met een `&` ("ampersand") bij de vijf parameters van de functies.
**d.** Als in de functie `gotye` staat `b += kimbra (( i < b ),a,b);`, compileert het programma dan nog? Onderscheid de gevallen met en zonder `&`, zie **b** en **c**.
**e.** Vervang `b += kimbra (jn,a,b);` door `a += kimbra (jn,a,b);`. Wat gaat er mis in de werking van het programma? Of de `&`'s erbij staan maakt daarbij overigens niet uit.

**3.** Gegeven is een `m` bij `n` (beide `const` $> 0$) Booleaans array `P`. Hierbij geeft `P[i][j]` de kleur van pixel $(i, j)$ aan, waar `true` (`T`) staat voor zwart en `false` (`F`) voor wit.

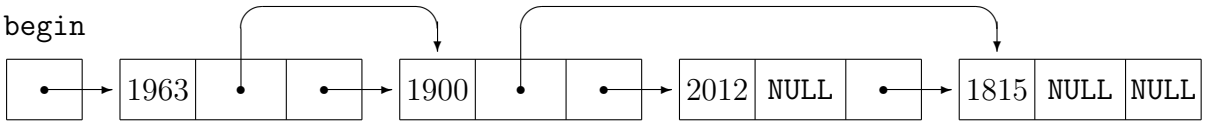
```
T F T T T F
F F T F F F
F T F F T T
```

**a.** Schrijf een C++-functie `inverteer (P)` die `P` inverteert: alle witte pixels moeten zwart worden, en alle zwarte wit.
**b.** Schrijf een C++-functie `int hoogste (P,j)` die in `j` de kolom-index van de kolom met de meeste zwarte punten (pixels) van `P` oplevert, en dit aantal retourneert. Als er meerdere kolommen dit maximale aantal realiseren, moet de grootste index van de betreffende kolommen in `j` terecht komen. Het aantal moet als geheel percentage van het aantal rijen (tussen 0 en 100), op de gebruikelijke wijze afgerond, worden opgeleverd. In het voorbeeld moet `j` dus 4 worden, en 67 (namelijk 2/3) worden teruggegeven.
**c.** Voor een Japanse puzzel staan bij elke rij in volgorde de lengtes van de (maximaal) aaneengesloten series zwarte punten, gescheiden door komma's. Naast `T T T F F T T F T F` staat 3,2,1. Schrijf een C++-functie `doerij (P,i)` die voor de `i`-de rij van `P` deze getallen afdrukt (met `cout << ...`; gebruik geen array om tussenresultaten op te slaan, druk een getal af zodra het bekend is). Neem aan dat $0 \leq$ `i` $<$ `m`, en verder dat er minstens één `true` in rij `i` staat. Er moeten alleen komma's *tussen* getallen komen. In het voorbeeld voor rij 0: 1,3; voor rij 1: 1; en voor rij 2: 1,2.

**4.** Gegeven is het volgende type:

```
class jaar { public: int jaartal; jaar* volg2; jaar* volg; };
```

Met behulp hiervan worden lijstjes met jaartallen opgebouwd. Het veld `volg` bevat een pointer naar het volgende `jaar`-object. De `volg2`-pointer (de middelste vakjes in het voorbeeld) wijst bij even jaartallen naar het *twee* verderop gelegen `jaar`-object, en bij oneven jaartallen naar het volgende (soms `NULL`). Een voorbeeld (`begin` van type `jaar*`):




**a.** Schrijf een C++-functie `verwijder (begin)` die het eerste `jaar`-object uit de lijst (met `begin` van type `jaar*` als ingang) netjes verwijdert, mits dat object bestaat en het jaartal erin niet gelijk is aan 2012.
**b.** Schrijf een C++-functie `verwissel (begin)` die de jaartallen uit de eerste twee `jaar`-objecten uit de lijst met ingang `begin` verwisselt, mits deze objecten bestaan, het eerste jaartal oneven is en het tweede jaartal even is. Maak ook de `volg2`-pointers in orde!
**c.** Schrijf een C++-functie `voegtoe (begin,jr)` die een nieuw `jaar`-object met jaartal `jr` erin vooraan de lijst met ingang `begin` toevoegt. Denk aan de `volg2`-pointer.
**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.
**e.** Schrijf een C++-functie `int laatste (begin)` die het laatste jaartal uit de lijst met ingang `begin` oplevert. Als de lijst leeg is, moet $-1$ worden teruggegeven. Er mag maximaal één keer een `volg`-pointer gebruikt worden.