Tentamen Programmeermethoden
Maandag 6 januari 2014, 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) gehele getallen.
a. Schrijf een C++-functie `hoevaak (A,X,n)` die teruggeeft hoe vaak het gehele getal `X` in het array `A` voorkomt.
b. Schrijf een Booleaanse C++-functie `uniek (A,n)` die precies dan `true` teruggeeft als geen enkel getal twee maal (of vaker) voorkomt in `A`, en anders `false`. Hierbij moet de functie van **a** *zinvol* gebruikt worden (hoe vaak komt `A[i]` voor?).
c. Schrijf een C++-functie `meest (A,n)` die het meest voorkomende getal uit `A` teruggeeft. Als er verschillende kandidaten zijn (bijvoorbeeld voor het array `17 12 30 12 42 30`) moet het kleinste getal dat het meest voorkomt worden geretourneerd. In het voorbeeld is dit `12` (dat even vaak voorkomt als `30`). Maak opnieuw gebruik van de functie van **a**.
d. Schrijf een C++-functie `sorteer (A,n)` die de getallen in `A` zodanig ordent dat voor alle getallen (behalve het laatste) geldt dat ze hooguit even vaak voorkomen als hun rechter buurman. Tip: pas de C++-code voor *bubblesort* eenvoudig aan; gebruik **a**.
e. Hoe vaak wordt de functie `hoevaak` aangeroepen in **d**?

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 tommy (int n, int s) {
  for ( a = 1; a <= n; a++ ) {
    s += a; cout << a << "," << n << "," << s << endl; }//for
  return s;
}//tommy
int barbara (int x, int y) {
  if ( x > y ) y = tommy (x-y,x); else x = tommy (y-x,y);
  cout << x << "," << y << endl; return x+y;
}//barbara
```

Verder zijn de globale variabelen `a` en `b` gegeven (van type `int`). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit; wat is de waarde van `a` na de for-loop?):

```
a = 5; b = 2; b = barbara (a,b); cout << a << "," << b << endl;
```

c. Als **b**, maar nu met een `&` (“ampersand”) bij de parameters `s`, `x` en `y` van de functies.
d. Als **c**, dus met drie `&`'s, maar nu voor:

```
a = 5; b = 2; b = barbara (b,b); cout << a << "," << b << endl;
```

e. Geef een eenvoudige uitdrukking voor de return-waarde van een aanroep `tommy (n,s)`, uitgedrukt in `n` en `s`. Zet hierbij geen `&` bij de parameter `s`.
f. Als in de functie `tommy` ergens `s = barbara (n,2*n);` staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `H` met verschillende gehele getallen ≥ 0. Hierbij geeft `H[i][j]` de hoogte in een sneeuwlandschap ter plekke $(\texttt{i}, \texttt{j})$ aan. De constantes `m` en `n` hoeven bij deze opgave niet doorgegeven te worden als parameter.

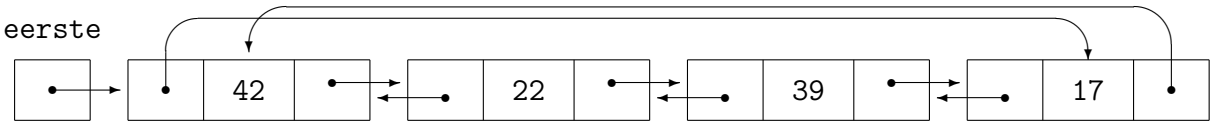
```
45 42 40 58
67 33 31 99
87 88 24 22
11 62 51 39
```

a. Er is sprake van een *horizontale glijbaan* van plek A naar plek B, als de plekken verschillen, in dezelfde rij zitten, waarbij B rechts van A ligt, en de hoogtes van A naar B dalen. Schrijf een Booleaanse C++-functie `horbaan (H,Ax,Ay,Bx,By)` die bepaalt of er zo'n glijbaan is van $A = (\texttt{Ax}, \texttt{Ay})$ naar $B = (\texttt{Bx}, \texttt{By})$. Neem aan dat $0 \leq \texttt{Ax}, \texttt{Bx} < \texttt{m}$ en $0 \leq \texttt{Ay}, \texttt{By} < \texttt{n}$. Van $(0, 1)$ naar $(0, 2)$ geeft `true`, naar $(0, 3)$ `false`.
b. Een horizontale glijbaan heet *maximaal* als hij niet naar rechts kan worden uitgebreid met een nieuwe plek. Schrijf een Booleaanse C++-functie `maxhorbaan (H,Ax,Ay,Bx,By)` die precies dan `true` teruggeeft als er een maximale horizontale glijbaan is van $A = (\texttt{Ax}, \texttt{Ay})$ naar $B = (\texttt{Bx}, \texttt{By})$. Gebruik **a**. Van $(0, 0)$ naar $(0, 1)$ geeft `false`, naar $(0, 2)$ `true`.
c. Schrijf een C++-functie `aantal (H,Ax,Ay)` die bepaalt hoeveel plekken vanuit $A = (\texttt{Ax}, \texttt{Ay})$ gepasseerd worden door afwisselend maximale horizontale glijbanen naar rechts en maximale verticale glijbanen (analoog gedefinieerd) naar onder te nemen, zolang het kan. Alle plekken, ook tussenliggende, van de gebruikte glijbanen tellen mee, evenals A zelf. Tip: gebruik *niet* de functies van **a** of **b**. Ga zo ver mogelijk naar rechts, naar onder, naar rechts, enz. In het voorbeeld levert `aantal (H,0,0)` via 45-42-40-31-24-22 op: 6.

4. Gegeven is het volgende type:

```
class klant { public: int nummer; klant* opvolger; klant* voorganger; };
```

Hiermee wordt een "cirkelvormige" lijst van klanten gemaakt. Het veld `opvolger` bevat een pointer naar het volgende `klant`-object, het veld `voorganger` uiteraard naar het vorige. Hierbij is de voorganger van de eerste klant de laatste, en de opvolger van de laatste is de eerste. Een voorbeeld (`eerste` van type `klant*`):



a. Schrijf een C++-functie `verwijder (eerste)` die het `klant`-object uit de structuur dat door `eerste` van type `klant*` wordt aangewezen, netjes verwijdert. Neem aan dat er oorspronkelijk minstens twee klanten zijn.
b. Schrijf een C++-functie `voegtoe (eerste,klantnr)` die een nieuw `klant`-object met `klantnr` erin in de niet-lege structuur met ingang `eerste` toevoegt. De pointer `eerste` moet naar de nieuwe klant gaan wijzen.
c. Schrijf een C++-functie `wissel (eerste)` die de klantnummers van de twee voorste klanten omwisselt, mits deze getallen verschillen en beide even zijn (zoals in het voorbeeld). Controleer of de lijst niet leeg is.
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? Mag het, moet het? Leg duidelijk uit.
e. Schrijf een C++-functie `draaiom (eerste)` die alle pointers in de (eventueel lege) lijst omdraait: elke `voorganger`-pointer moet naar de oorspronkelijk volgende klant gaan wijzen, en elke `opvolger`-pointer moet naar de oorspronkelijk vorige klant gaan wijzen.