Tentamen Programmeermethoden
Maandag 5 januari 2009, 14.00–17.00 uur
Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen (constanten uitgezonderd) in de heading of als locale variabele voorkomen; vul zelf de headings goed in. De opgaven tellen alle vier even zwaar mee. Succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/res08.txt`.

1. In een array `int A[n]` stoppen we `n` (een `const` > 0) gehele positieve (> 0) getallen.
a. Schrijf een C++-functie `sort (A,n)` die `A` oplopend sorteert met behulp van *bubblesort*.
b. Neem aan dat `A` gesorteerd is. Schrijf een efficiënte C++-functie `bool dubbel (A,n)` die precies dan `true` oplevert als er minstens één getal twee maal of vaker in `A` voorkomt.
c. Neem aan dat `A` gesorteerd is en geen dubbelen bevat: het zijn postzegel(waarde)s. We zoeken naar postzegels waarvan de waardes precies sommeren tot een gegeven `w`. Schrijf een C++-functie `int plak (w,A,n,i1,i2)` die aangeeft met *hoeveel* postzegels `w` kan worden "geplakt". Er mogen *maximaal twee* zegels (eventueel van dezelfde waarde; in totaal zo weinig mogelijk) gebruikt worden; als het kan, komen mogelijke *array-indices* in `i1` en (als het er twee zijn) `i2`; anders wordt `i2`: -1. Als `A` waardes `1 4 7` bevat (`n` $= 3$), en `w` $= 8$, is 2 het antwoord, en `i1` $=$ `i2` $= 1$ ($4 + 4 = 8$) of (kies zelf) `i1` $= 0$ en `i2` $= 2$ ($1 + 7 = 8$). Als `w` niet geplakt kan worden, is 0 het antwoord (`w` $= 6$), en `i1` $=$ `i2` $= -1$.
d. Schrijf een C++-functie `int serie (A,n)` die de lengte `max` (≥ 0) bepaalt van de langste serie $1, 2, \ldots,$ `max` waarvan elke waarde geplakt kan worden met **c**'s methode.

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 xyz (bool twee, int x) {
  int i = 42; twee = ! twee;
  for ( i = x; i <= z; i++ ) { y++; x++; i++; z--; }//for
  cout << i << "," << x << "," << y << "," << z << "," << (int)twee << endl;
  return ( twee && ( i > z ) ); }//xyz
int abcd (bool een, int x) {
  int y = 42; een = ! een;
  if ( een || ! xyz (een,x) ) { een = ( y > z ); if ( een ) z = 42; }//if
  else { y = z; x = x + y; z = x - y; een = ( y > x ); }//else
  return z; }//abcd
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`) en een `bool` `b`. Voordat de functie `abcd` wordt aangeroepen hebben zij de waarde `1`, `12`, `6` en `true`, respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit; let op het gebruik van *casting*: `(int)b`, oftewel `static_cast<int>(b)`):

```
cout << abcd (b,x) << endl; cout << x << y << z << (int)b << endl;
```

c. Als **b**, maar nu met een `&` bij de vier parameters van de functies.
d. Als **b**, dus zonder de vier `&`'s, maar nu met aanroep

```
cout << abcd ((bool)(y-2*z),x) << endl; cout << x << y << z << endl;
```

e. Als **d**, maar nu weer met de vier `&`'s erbij.

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `bergen`, gevuld met verschillende gehele getallen ≥ 0; `bergen[i][j]` stelt de hoogte op positie (`i`, `j`) voor.

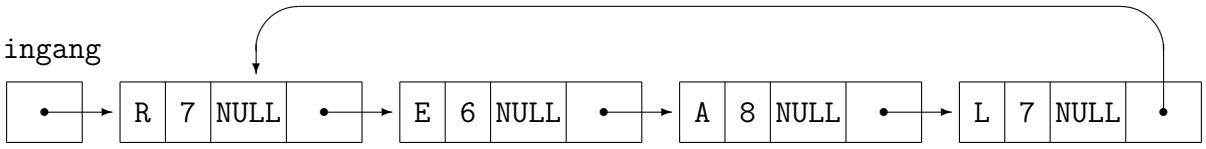
```
19 10  9  5 21
 3 16 12 13  2
22 17 14 18  8
```

a. Schrijf een C++-functie `int hoogste (bergen,i,j)` die in `i` en `j` de coordinaten van het hoogste punt in `bergen` oplevert, en de bijbehorende hoogte teruggeeft. In het voorbeeld: $(2, 0)$ met hoog(s)te 22.
b. Vanuit een gegeven punt kun je horizontaal en verticaal kijken *tot* het eerstvolgende hoger gelegen punt — of tot aan de rand van het array. Schrijf nu een C++-functie `int zien (bergen,i,j)` die bepaalt hoeveel punten je vanuit een gegeven (`i`, `j`) kunt zien ($0 \leq$ `i` $<$ `m`, $0 \leq$ `j` $<$ `n`). In het voorbeeld, vanuit positie $(1, 3)$ met hoogte 13, kun je 4 punten zien, namelijk die met hoogtes 5, 12, 13 (jezelf) en 2.
c. Schrijf een C++-functie `int puttop (bergen)` die retourneert hoeveel punten er zijn waar je niets kunt zien (alleen jezelf) of juist alles (in eigen rij en kolom). In het voorbeeld 5 stuks: 5, 21, 3, 2 en 22. Gebruik **b**.

4. Gegeven is het volgende type:

```
class kand { public: char naam; int cijfer; kand* reserve; kand* volg; };
```

Met behulp hiervan worden rijtjes (lijstjes) met namen van kandidaten (hoofdletters) en hun cijfer opgebouwd. Het veld `volg` bevat een pointer naar het volgende `kand`-object, waarbij in de laatste een pointer naar de eerste zit — en dus naar zichzelf als er maar één element is. De `reserve`-pointer is voorlopig steeds `NULL`. Een voorbeeld (`ingang` van type `kand*`):



a. Schrijf een C++-functie `verwissel (ingang)` die de `naam`-inhouden van de twee eerste objecten — indien aanwezig — in de alfabetisch juiste volgorde zet. In het voorbeeld moeten de `R` en de `E` verwisseld worden. De `cijfer`-velden moeten in zo'n geval ook worden meegewisseld.
b. Schrijf een C++-functie `voegtoe (ingang,nm,cf,laatste)` die een nieuwe kandidaat met naam `nm` en cijfer `cf` erin vooraan de structuur (met `ingang` van type `kand*` als ingang) toevoegt. Neem aan dat de pointer `laatste` van type `kand*` naar de "laatste" kandidaat wijst, of `NULL` is — als de lijst leeg was; gebruik en update deze zonodig.
c. Schrijf een C++-functie `verwijder (ingang,laatste)` die de eerste kandidaat uit de lijst (met `ingang` van type `kand*` als ingang) verwijdert, mits deze een cijfer ≤ 5 heeft. Denk aan de lege lijst. Gebruik en update zonodig weer `laatste` (zie **b**).
d. In de functies bij **a**, **b** en **c** staan in de heading een of twee pointers. 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 `kand* delaatste (ingang)` die een pointer naar de "laatste" kandidaat oplevert (eventueel `NULL`), en alle `reserve`-velden naar deze laatste kandidaat laat wijzen.