Tentamen Programmeermethoden
Maandag 3 augustus 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. We hebben een array `A` (`double A[n]`, met `const int n = 12345;`) met `n` verschillende getallen.
a. Geef een C++-functie `void wissel (A,i,j)` die de waardes van de array-elementen `A[i]` en `A[j]` verwisselt (bij vaste `i` en `j` met $0 \leq$ `i`, `j` $\leq$ `n-1`).
b. Geef een C++-functie `void bubblesort (A,n)` die het array `A` oplopend sorteert met *bubblesort*. De eenvoudigste versie is goed.
c. Geef een C++-functie `double kde (A,k)` die het `k`-de element in grootte (met `1` $\leq$ `k` $\leq$ `n`) van het array `A` oplevert. Dat is array-element `A[n-k]` *als* het array gesorteerd zou zijn. Pas daarom de functie van **b** zodanig aan dat na de `k`-de doorgang het sorteren afbreekt, en het antwoord gegeven wordt.
d. Hoeveel vergelijkingen tussen `double`'s doet het algoritme van **c** voor algemene `n` en `k`? (In het antwoord mag een sommatie blijven staan.)
e. Als **c**, maar nu voor een situatie waarin het array `A` gesorteerd is, waarna twee willekeurige elementen zijn verwisseld. Er mogen maximaal `n-1` vergelijkingen tussen `double`'s gedaan worden.

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 heb je ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.
b. Een zeker C++-programma bevat de volgende programmaregels:

```
int jeroen (int x) {
  g++; x = x + g; cout << g << x << endl; return x;
}//jeroen
int paul (int x, int y) {
  int ik = jeroen (x), i;
  for ( i = 1; i <= ik; i++ ) y += jeroen (y);
  cout << g << x << y << ik << i << endl; return y;
}//paul
```

Verder zijn de globale variabelen `g`, `a` en `b` gegeven (alle van type `int`). Voordat de functie `paul` wordt aangeroepen hebben zij de waarde `0`, `3` en `4`, respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
cout << paul (a,b) << endl; cout << g << a << b << endl;
```

c. Als **b**, maar met drie `&`'s in de headings van `paul` en `jeroen`.
d. Idem, als bij **c**, dus met de drie `&`'s erbij, maar nu voor

```
cout << paul (g,g) << endl; cout << g << a << b << endl;
```

e. Stel je wilt in de functie `jeroen` de functie `paul` aanroepen. Mag dat, en zo nee, wat moet je doen om het wel te mogen laten kunnen? En is er dan sprake van recursie?

3. Deze opgave gaat over `n` bij `n` arrays met gehele getallen, groter dan of gelijk aan 0 en kleiner dan n^3: `int A[n][n]`. Getallen (behalve eventueel 0) komen maximaal één keer voor. Enkele voorbeelden met `n` gelijk aan 3:

```
2   3  15        0   0   0        0   0   0
5  12   8        0  12   8        0   0   8
0   7   9        0   7   9        0   0   0
```

a. Schrijf een C++-functie `int klein (A,i,j)` die het kleinste array-element (ongelijk aan 0) uit `A` retourneert. Rij- en kolom-index van dit getal moeten via `i` en `j` worden teruggegeven. Als het array geheel uit nullen bestaat moet n^3 worden teruggegeven; de waardes van `i` en `j` doen er dan niet toe. Voorbeeld linker array: `i` en `j` worden `0`, retourneer `2`; middelste array: `i` wordt `2`, `j` wordt `1`, retourneer `7`.
b. Schrijf een C++-functie `void maak0 (A)` die de rij en kolom ven het kleinste getal ongelijk nul uit `A` met nullen vult. Gebruik de functie van **a**. Als het array geheel uit nullen bestaat doet de functie niets. Voorbeeld: van linker array naar middelste, of van middelste naar rechter.
c. Schrijf een C++-functie `int over (A)` die het kleinste getal ongelijk nul teruggeeft dat in `A` overblijft als je vaak genoeg de functie van **b** aanroept (dus als het ware vlak voordat alle getallen nul worden). Als het originele array geheel uit nullen bestaat moet de functie n^3 retourneren. Voor alle drie voorbeeld-arrays: `8`.

4. Gegeven is:

```
class Mens { public: char naam;  // naam van de persoon
                     int hoever;      // verweg wijst hoever verder
                     Mens* verweg;    // wijst persoon verderop aan
                     Mens* volgende;  // wijst volgende persoon aan
};//Mens
```

Met behulp hiervan kan een enkelverbonden lijst van mensen worden opgebouwd. Zo'n lijst bestaat uit vakjes met een `char` en een `int`, en twee pointers naar respectievelijk een zich `hoever` vakjes (minstens `0`) verder in de lijst bevindende persoon en de volgende persoon. Als `hoever = 0` wijst `verweg` het vakje zelf aan.
a. Voor zowel de eerste als de tweede persoon van een niet-lege lijst met ingang `groep` van type `Mens*` wordt zijn/haar naam van hoofdletter naar kleine letter omgezet, en andersom. Neem aan dat de naam een kleine letter of hoofdletter is. Schrijf een C++-functie `hOOFD (groep)` die dit doet. Vergeet niet de situatie met slechts één persoon.
b. Schrijf een C++-functie `verwijder (groep)` die de eerste persoon uit de lijst `groep` verwijdert. Als de lijst leeg is of als de door `verweg` aangewezen persoon hetzelfde heet als de eerste persoon, hoeft er niets te worden gedaan.
c. Schrijf een C++-functie `voegtoe (groep,naampje,afstand)` die een nieuwe persoon, geheten `naampje` (van type `char`), toevoegt vooraan de eventueel lege lijst `groep` (van type `Mens*`). Geef ook het `verweg`-veld de juiste waarde, aannemend dat `afstand` (= `hoever`) hooguit `2` is, en dat de betreffende persoon bestaat in de lijst.
d. In de functies bij **a**, **b** en **c** staat in de heading 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 `bool` C++-functie `controle (groep)` die controleert of voor alle personen in de lijst met ingang `groep` het `verweg`-veld correct de persoon `hoever` verderop aanwijst.