Tentamen Programmeermethoden
Vrijdag 31 maart 2006, 10.00–13.00 uur
Universiteit Leiden — Informatica

Bij alle te schrijven functies moeten de variabelen in de heading of als locale variabele voorkomen; vul zelf de headings goed in. De opgaven tellen alle vier even zwaar mee. Veel succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/res05.txt`.

1. a. We willen array `A` met `n` `double`'s oplopend sorteren. Geef C++-code voor *bubblesort*.
b. Stel dat de getallen in `A` oplopend gesorteerd zijn. Iemand halveert één door hem gekozen (maar voor ons onbekend) array-element. Stel dat het resulterende array nu niet meer gesorteerd is. Schrijf een C++-functie `void herstel1 (A,n)` die *het originele array* weer herstelt door het gehalveerde element te verdubbelen.
c. Idem, maar nu moet de functie `void herstel2 (A,n)` *de sortering* efficiënt herstellen, door het gehalveerde element naar zijn juiste plaats te brengen.
d. Stel dat na het halveren van één element het array nog steeds gesorteerd is. Er is nu gegeven dat het mogelijk is het originele array op unieke wijze hieruit te reconstrueren, en dat het gehalveerde element *niet* het laatste array-element is. Schrijf `void herstel3 (A,n)` die weer *het originele array* herstelt. Hint: in `3.0 4.0 5.0 12.0 20.0` moet `5.0` wel de boosdoener zijn.

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 een (int x, int y ) {
   int z = 4;
   if ( x > y ) x = x - y + z; else y = y + x + z;
   z += x; x = x + y; y = y + 5;
   cout << x << ", " << y << ", " << z << endl;
   return (z+y+x)/5;
}//een
void twee (int a, int b, int c) {
   z = 4; x = 3;
   if ( een (a,b) < c ) { c = c + a; a -= x; b += x; x++; }
   else { c = 42; a++; b += 2; x--; }
   cout << a << ", " << b << ", " << c << endl;
}//twee
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `twee` wordt aangeroepen hebben zij de waarde 31, 3 en 2006 respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

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

c. Als **b**, maar nu staat er een `&` bij de twee parameters van `een`.
d. Als **c**, maar nu staat er ook een `&` bij alle parameters van de functie `twee`.
e. Je wilt nu binnen de functie `een` de functie `twee` aanroepen. Dit is dan (indirecte) recursie. Wat moet je dan nog toevoegen in je programma en waarop moet je letten als het gaat om de werking van het programma?

3. We hebben een array `int C[n][n]` (met constante `n`) met gehele getallen.
a. Een *vierkant* bestaat uit een viertal array-elementen (de *hoekpunten*) waarbij de array-elementen uit precies 2 rijen en 2 kolommen komen (afstand tussen de rijen gelijk aan afstand tussen de kolommen; bijvoorbeeld `C[1][2]`, `C[1][6]`, `C[5][2]` en `C[5][6]`; zie ook **b**). Schrijf een C++-functie `int vier (C)` die bepaalt hoeveel vierkanten `C` heeft met in alle vier de hoekpunten dezelfde waarde.
b. Neem vanaf nu aan dat alle array-elementen verschillen.
Schrijf een C++-functie `void krimp (C,i,j,b)` die het vierkant met linksboven `C[i][j]` en breedte `b` (minstens `1`; rechtsboven zit `C[i][j+b]`) één in breedte (en hoogte) laat krimpen, en wel als volgt: zoek het hoekpunt met de grootste waarde; dit blijft vast; de twee zijden waarop dit hoekpunt ligt worden 1 korter. Het nieuwe vierkant ligt binnen het oude. De functie moet de waarden van `i`, `j` en `b` aanpassen zoals beschreven.
c. Schrijf een C++-functie `int pers (C)` die, beginnend met het grootste vierkant, bestaande uit de hoekpunten van `C` zelf, herhaald krimpt. De functie moet stoppen zodra het hoekpunt linksboven vast blijft, of als het vierkant breedte 0 heeft. Het aantal "krimps" moet worden geretourneerd.

4. Gegeven is het volgende type:

```
class info { public:
  info* volgende; info* andere;
  int aantal;                  };//info
```

Met behulp hiervan worden rijtjes (lijstjes) opgebouwd. Het veld `volgende` bevat steeds een pointer naar het direct "rechts" ernaast gelegen vakje, het veld `andere` bevat een pointer naar een lijstje met precies `aantal` vakjes (0 als de pointer `NULL` is). In dat lijstje zijn de `andere`-pointers steeds `NULL`, en kunnen de `aantal`-velden voor andere informatie gebruikt worden. Een voorbeeld (`ingang` van type `info*`):

```
ingang ---> 2 | ---> 0 NULL ---> 0 NULL ---> 0 NULL ---> 1 | NULL
              |                                            |
              ---> 12 NULL ---> 4 NULL NULL                ---> 8 NULL NULL
```

Het vakje met `2` erin heeft hier nog een pointer `andere` naar het vakje met `12` erin, waarachter nog een vakje met `4` is.
a. Schrijf een C++-functie `voegtoe (ingang)` die een nieuw vakje vooraan de structuur (met `ingang` van type `info*` als ingang) toevoegt. Hierbij moet het `aantal`-veld `0` worden.
b. Schrijf een C++-functie `verwijder (ingang)` die het eerste `info`-vakje uit de lijst (met `ingang` van type `info*` als ingang) verwijdert, compleet met het eventuele vakje eronder. Neem aan dat het `aantal`-veld van het eerste vakje maximaal `1` is. Denk aan de lege lijst.
c. Schrijf een C++-functie `vulaan (ingang, getal)` die een nieuw vakje (met `aantal`-veld gelijk aan `getal`) aan het `andere`-veld van het ingangs-vakje koppelt — indien er een eerste vakje is, en diens `aantal`-veld `0` is.
d. In de functies bij **a**, **b** en **c** staat in de heading de parameter `ingang`. 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 ook uit wat er bij deze twee mogelijkheden precies gebeurt tijdens executie van de betreffende functie.
e. Schrijf een C++-functie `int lengte (ingang)` die voor het algemene geval uitrekent wat het totaal aantal gebruikte `info`-vakjes is; voor de voorbeeldstructuur zijn dat er `8`. Voor de lege lijst moet de functie `0` opleveren.