Tentamen Programmeermethoden
Maandag 4 augustus 2008, 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/res07.txt`.

1. In een array `int A[n]` stoppen we `n` (een `const` > 0) positieve en negatieve getallen, ongelijk aan 0.
a. Schrijf een C++-functie `int desom (A,i,j)` die de som van de array-elementen `A[i]`, `A[i+1]`, ..., `A[j]` berekent. Neem aan dat $0 \leq$ `i` $\leq$ `j` $<$ `n`.
b. Schrijf een C++-functie `int grootsom (A,n)` die de grootste som van niet-lege rijtjes opeenvolgende array-elementen uit `A` teruggeeft. Gebruik **a**.
c. Geef een C++-functie `sorteer (A,n)` die `A` oplopend sorteert met behulp van *bubblesort*. Zorg er hierbij voor dat als bij een rondgang door het array geen verwisselingen plaats hadden, het algoritme stopt.
d. Doe opnieuw onderdeel **b**, maar neem nu aan dat de negatieve getallen voor de positieve getallen staan. Geef een efficiënte oplossing. Neem aan dat `A` minstens één positief getal bevat.
e. Als onderdeel **d**, maar nu is er niets bekend over het aantal positieve getallen. Wel staan nog steeds de negatieve voor de positieve.

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 dat de volgende functies bevat:

```
int bas (int a, int b) {
  a = a + b; b = a - b; a = a - b;
  cout << a << "," << b << endl; return a;
}//bas
void aad (int b, int a) {
  int i; for ( i = 1; i <= a; i++ ) { x += bas (a,b); }//for
  cout << a << "," << b << "," << i << "," << x << endl;
}//aad
```

Laten verder de globale variabelen `x`, `y` en `z` (alle van type `int`) gegeven zijn. Voor aanroep van de functie `aad` hebben zij de waarden 1, 3 en 5 respectievelijk. Wat is dan de uitvoer van (leg je antwoord duidelijk uit):

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

c. Dezelfde vraag als **b**, maar nu staat bij *elke* parameter een `&` (vier maal dus).
d. Wat gebeurt er bij de aanroep `aad (x,x)`, weer in het geval van **c**?
e. Wat is in het algemeen de waarde die in `y` en `z` zit na aanroep `bas (y,z)`, uitgedrukt in de oorspronkelijke `y`, `z` en `x`? (Wederom met de `&`'s erbij.) Evenzo voor `aad (y,z)`.

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `letters`, gevuld met hoofdletters.

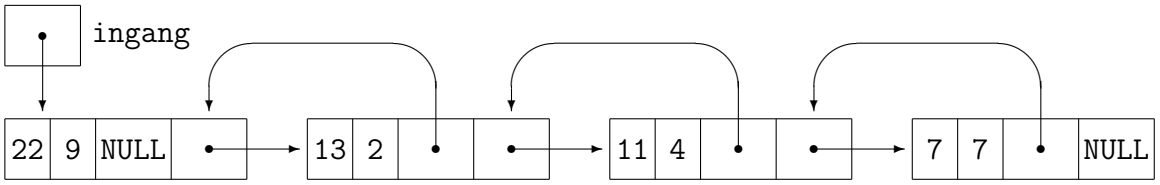
```
G H F C      B C K
C I G U      C D N
G I M R      J Q Q
```

a. Schrijf een functie `bool controle (letters,i)` die controleert of de `i`-de rij uit `letters` oplopend is gesorteerd. De derde rij (`i` $= 2$) van het linker voorbeeld is dat wel, de eerste en tweede niet.
b. Schrijf een C++-functie `bool zoek (letters,i,j,let)` die de letter `let` opzoekt in `letters`, en zo mogelijk indices `i` en `j` met `letters[i][j]` gelijk aan `let` oplevert. In dat geval moet `true` worden geretourneerd, anders `false`. Het algoritme moet stoppen zodra `let` gevonden wordt, ook als `let` meerdere keren voorkomt.
c. Neem nu aan dat zowel rijen als kolommen van `letters` oplopend gesorteerd zijn (zie het rechter voorbeeld). Schrijf opnieuw de functie van **b**, maar maak gebruik van de sortering, en wel als volgt. Stel je bent op plek (`i`, `j`), en je weet dat `let` niet voorkomt in de rijen tot en met rij `i-1` en de kolommen tot en met kolom `j-1`. Controleer nu de eventuele rest van rij `i` en kolom `j`, en hoog deze beide met 1 op. Let er op niet uit het array te vallen.

4. Gegeven is het volgende type:

```
class info { public: info* volg; info* vorig; int som; int getal; };
```

Met behulp hiervan worden rijtjes (lijstjes) met getal-getal combinaties opgebouwd. Het veld `volg` bevat een pointer naar het *volgende* object in de lijst (of `NULL`), `vorig` bevat een pointer naar het *vorige* object (of, bij het eerste object, `NULL`). Het `som`-veld moet de som van alle `getal`-velden vanaf (en inclusief) het huidige object bevatten. Een voorbeeld (`ingang` van type `info*`;), waarbij `volg` de meest rechtse pointer in ieder object is (bijvoorbeeld, 13 is $2 + 4 + 7$):



a. Schrijf een C++-functie `voegtoe (ingang,get)` die een nieuw object met getal `get` erin vooraan de structuur (met `ingang` van type `info*` als ingang) toevoegt. Denk ook aan de `vorig`-pointers (mits de originele lijst minstens één object had). En geef het `som`-veld de juiste waarde.
b. Schrijf een C++-functie `verwijder (ingang)` die het eerste object uit de lijst (met `ingang` van type `info*` als ingang) verwijdert indien in dat object alleen oneven getallen zitten (in `som`- en `getal`-veld). Denk aan de lege lijst, en een eventuele `vorig`-pointer die `NULL` moet worden.
c. Schrijf een C++-functie `verwissel (ingang)` die de `getal`-velden uit eerste en tweede object verwisselt (dus de inhouden), indien deze bestaan, en anders niets doet. De `som`-waarden moeten ook in orde gemaakt worden.
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 duidelijk uit.
e. Schrijf een C++-functie `repareer (ingang)` die alle "kapotte" `vorig`-pointers repareert. Een `vorig`-pointer heet kapot als hij niet naar zijn voorganger wijst; en uiteraard moet de eerste `vorig`-pointer `NULL` zijn.