Tentamen Programmeermethoden
Vrijdag 28 maart 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 gehele getallen (≥ 0).
a. Schrijf een C++-functie `int komtvoor (A,x)` die bepaalt hoevaak het gehele getal `x` in `A` voorkomt.
b. Pas de functie van **a** zo aan dat — als `x` precies één keer voorkomt in `A` — in plaats van 1 nu de arrayindex `i` met `A[i]` = `x`, voorzien van een min, en met 1 eraf getrokken, wordt geretourneerd. Dus antwoord 3 betekent dat `x` precies 3 maal voorkomt, antwoord -3 dat `x` precies 1 maal voorkomt, en wel op positie 2. En 0 betekent dat `x` niet voorkomt.
c. Schijf een C++-functie `bool verschillend (A)` die `true` oplevert als alle getallen in `A` verschillen, en anders `false`. Het array mag alleen met de functie van **b** benaderd worden, waarbij de parameter `x` bij aanroep wel een array-element mag zijn.
d. Schijf een C++-functie `int grootste (A)` die het grootste getal uit `A` oplevert. Het array mag alleen met de functie van **a** of **b** benaderd worden.
e. Geef een C++-functie `sorteer (A)` die `A` oplopend sorteert met behulp van *selection sort* = *simpelsort*: zet herhaald het kleinste getal "vooraan".

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:

```
void paul (int a, int b) {
  int i; int x = 20; for ( i = 1; i <= a; i++ ) { b++; x--; }//for
  cout << a << "," << b << "," << i << "," << x << endl;
}//paul
void david (int b, int a) {
  int i; for ( i = 1; i <= x; i++ ) { paul (a, b); x--; }//for
  a--; cout << a << "," << b << "," << i << "," << x << endl;
}//david
```

Laten verder de globale variabelen `x`, `y` en `z` (alle van type `int`) gegeven zijn. Voor aanroep van de functie `david` hebben zij de waarden 3, 4 en 3 respectievelijk. Wat is dan de uitvoer van (leg je antwoord duidelijk uit):

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

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

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

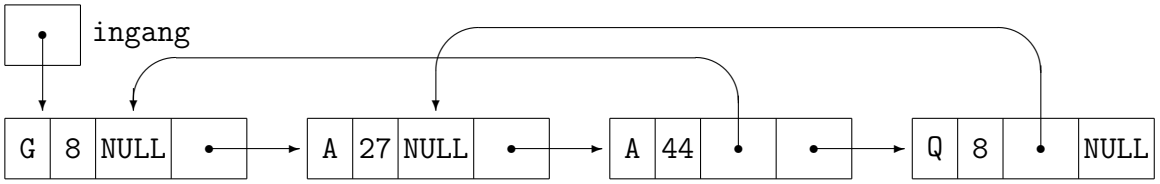
```
G H F C      X U L
C I G U      A I K
H W W H      A A O
```

a. Schrijf een functie `bool controle (puzzel,i)` die controleert of de `i`-de rij uit `puzzel` een *palindroom* is, dat wil zeggen van voor naar achter hetzelfde als van achter naar voor. De derde rij (`i = 2`) van het linker voorbeeld is een palindroom.
b. Schrijf een C++-functie `int telze (puzzel)` die telt hoevaak er twee klinkers (A/E/I/O/U) boven elkaar staan. Het linker voorbeeld: 0 (nooit); rechts: 3.
c. Schrijf een C++-functie `bool ja (puzzel,woord,w)` die bepaalt of een gegeven `woord`, een `char` array met `w` > 0 elementen, voorkomt in `puzzel`. Hierbij mag je, beginnend in een vakje naar keuze, herhaald horizontaal of verticaal naar een buurman lopen. Neem aan dat elk array-element in `puzzel` *verschillende* buren heeft. Zo komt in de linker puzzel `GHICGHF` voor, dus geeft de functie `true`. Alle mogelijke beginvakjes moeten geprobeerd worden.

4. Gegeven is het volgende type:

```
class info { public: info* volg; info* terug; char letter; int getal; };
```

Met behulp hiervan worden rijtjes (lijstjes) met letter-getal combinaties opgebouwd. Het veld `volg` bevat een pointer naar volgende object in de lijst (of `NULL`), `terug` bevat een pointer naar het *voorvorige* object (of, bij eerste en tweede object, `NULL`). Een voorbeeld (`ingang` van type `info*`;), waarbij `volg` de meest rechtse pointer in ieder object is:



a. Schrijf een C++-functie `voegtoe (ingang,let,get)` die een nieuw object met getal `get` en letter `let` erin vooraan de structuur (met `ingang` van type `info*` als ingang) toevoegt. Denk ook aan de `terug`-pointers (mits de originele lijst minstens twee objecten had).
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 vakje *niet* de letter `Q` *en niet* het getal `8` zitten. Denk aan de lege lijst, en een eventuele `terug`-pointer die `NULL` moet worden.
c. Schrijf een C++-functie `verwissel (ingang)` die eerste en tweede (via de `volg`-pointer) object verwisselt (dus *niet* de inhouden), indien deze bestaan en het getal uit het eerste object groter is dan dat uit het tweede, en anders niets doet. De `terug`-pointers hoeven niet goed gezet te 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 `info* voorste (info* pijl)` die een pointer naar het eerste object teruggeeft, gegeven een pointer `pijl` die een "willekeurig" object uit de lijst aanwijst. Neem aan dat de lijst minstens drie objecten heeft. De ingang van de lijst is dus niet rechtstreeks beschikbaar!