Tentamen Programmeermethoden
Maandag 8 januari 2007, 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/res06.txt`.

1. In deze opgave zijn alle gehele getallen > 0, en eindigen niet op een `0`. In een array `int A[n]` stoppen we `n` (een `const` > 0) gehele getallen.
a. Schrijf een C++-functie `int draaiom (x)` die het gehele getal `x` omgedraaid teruggeeft. Dus `12345` moet `54321` worden. Hint: benut herhaald `x % 10`, en deel `x` steeds door 10.
b. Schrijf een C++-functie `bool kleiner (x,y)` die precies dan `true` oplevert als het gehele getal `x` *lexicografisch* strikt kleiner is dan het gehele getal `y`. Dit betekent dat het eerste cijfer waarin `x` en `y` verschillen bepalend is voor de volgorde. Zo is `424189` lexicografisch kleiner dan `4247`, want `1` $<$ `7` (vierde cijfer). En `42` is lexicografisch kleiner dan `42142`, via `NIKS` $<$ `1` (derde cijfer). Als `x` $=$ `y` is de functiewaarde `false`. Hint: **a**.
c. Schrijf een C++-functie `int kleinste (A,n)` die de *array-index* van het lexicografisch kleinste element van `A` teruggeeft. Als dat element meerdere keren voorkomt, moet het de op een na kleinste array-index met de gevraagde eigenschap zijn.
d. Schrijf een C++-functie `void sorteer (A,n)` die `A` lexicografisch oplopend sorteert. Gebruik een sorteermethode naar keuze.

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 janpeter (int j, int grens) {
  j = j + grens; grens = j - grens; j = grens - j;
  cout << "Jan " << j << " en " << grens << endl; return (grens - j);
}//janpeter
int wouter (int i, int grens) {
  int som = grens;
  while ( i <= grens ) { som += janpeter (i,grens); i++; }//while
  cout << "Wout " << i << " en " << grens << endl; return som;
}//wouter
```

Verder zijn de globale variabelen `x`, `y` en `gr` gegeven (alle van type `int`). Voordat de functie `wouter` wordt aangeroepen hebben zij de waarde `0`, `1` en `5` respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = wouter (y,gr) - 3; cout << x << ", " << y << " en " << gr << endl;
```

c. Als **b**, maar nu staat er een `&` bij alle vier parameters.
d. Als **b** (dus *zonder* de vier `&`'s), en `x`, `y` en `gr` beginnen op `670`, `1` en `5`, voor

```
x = wouter (x,x) - 3; cout << x << ", " << y << " en " << gr << endl;
```

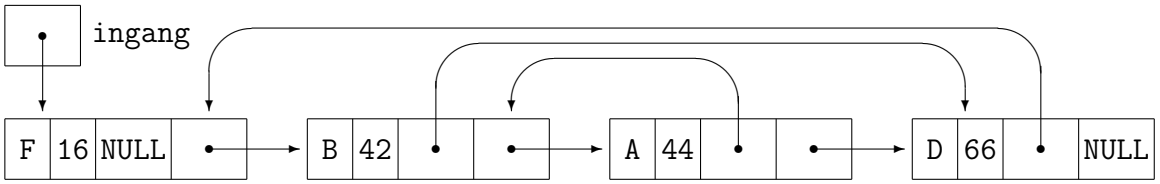
e. Je wilt nu binnen de functie `janpeter` de functie `wouter` aanroepen. Dit is (indirecte) recursie. Wat moet je dan nog toevoegen in je programma en waarop moet je letten als het gaat om de werking van het programma?

3. Gegeven is een `m` bij `n` (beide `const`) array `puz`, gevuld met gehele getallen. Elk getal > 0 stelt een speciaal schaakstuk voor. Zo staat `42` voor de (spring)dame, die alle vakjes in haar eigen rij, kolom en beide diagonalen waarin ze staat direct kan bereiken, waarbij we niet letten op eventuele stukken ertussen; en `0` staat voor een leeg vakje.
a. Schrijf een functie `bool slaan (puz,i,j,r,s)` die precies `true` is als een (denkbeeldige) dame op positie $(\texttt{i}, \texttt{j})$ in `puz` het vakje met positie $(\texttt{r}, \texttt{s})$ direct kan bereiken. De functie moet ook `false` opleveren als een of twee van de vakjes buiten het array liggen. Een dame bereikt niet direct het vakje waarop ze zelf staat.
b. Schrijf een C++-functie `int tel (puz,i,j,nr)` die alle *lege* vakjes die een dame vanuit positie $(\texttt{i}, \texttt{j})$ ($0 \leq \texttt{i} < \texttt{m}$ en $0 \leq \texttt{j} < \texttt{n}$) direct zou kunnen bereiken vult met het getal `nr`, en het betreffende aantal vakjes teruggeeft.
c. Neem nu aan dat op positie $(\texttt{i}, \texttt{j})$ een dame staat, en dat positie $(\texttt{r}, \texttt{s})$ leeg is. De dame mag in één zet naar een direct te bereiken leeg vakje. Schrijf een C++-functie `int kan (puz,i,j,r,s)` die het kleinste aantal (zetten) oplevert waarmee de dame positie $(\texttt{r}, \texttt{s})$ kan bereiken; geef `0` als dit helemaal niet kan. Tip: vul de in één stap te bereiken lege vakjes met `-1`, enzovoorts. Dat het array `puz` in de war raakt, doet er niet toe.

4. Gegeven is het volgende type:

```
class info { public: info* volgl; info* volgg; char letter; int getal; };
```

Met behulp hiervan worden rijtjes (lijstjes) met letter-getal combinaties opgebouwd; alle gebruikte letters en getallen verschillen onderling. Het veld `volgl` bevat een pointer naar het `info`-object met de eerstvolgende alfabetisch grotere letter (of `NULL`), het veld `volgg` een pointer naar het object met het eerstvolgende grotere getal (of `NULL`). Een voorbeeld (`ingang` van type `info*`; deze wijst steeds het object met het kleinste getal aan), waarbij `volgg` de meest rechtse pointer in ieder object is:



Het object met `B` en `42` erin heeft hier een pointer `volgl` naar het object met `D` en `66`, en een (horizontale) pointer `volgg` naar het object met `A` en `44`.
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. Neem aan dat het eerste getal uit de "oude" lijst groter is dan `get`. Maak de nieuwe `volgl`-pointer `NULL`, en verander (nog) niets aan de andere `volgl`-pointers.
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 `D` en het getal `66` zitten. Denk aan de lege lijst. Verander niets aan `volgl`-pointers.
c. Schrijf een C++-functie `verwissel (ingang)` die eerste en tweede (via de `volgg`-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.
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. Vul de functie `voegtoe` van **a** aan zodat na afloop alle `volgl`-pointers goed staan. Neem aan dat er minstens één letter $<$ `let` in de lijst voorkomt. Tip: zoek de grootste kleinere.