Tentamen Programmeermethoden
Donderdag 3 januari 2019
14:00–17:00 uur Informatica

Universiteit Leiden
The Netherlands

Bij alle functies moeten de variabelen (constanten uitgezonderd) in de heading of locaal voorkomen; vul zelf headings goed in. De te behalen punten (totaal 100) staan tussen haakjes bij de opgaven. Succes! Cijfers: te zijner tijd via Blackboard.

1. (25) In het array `int A[n]` staan `n` (een `const int` ≥ 2) verschillende positieve (> 0) gehele getallen.
a. (8) Schrijf een C++-functie `int klein (A,i,X,n)` die de kleinste waarde uit `A` teruggeeft die groter ($>$) is dan het meegegeven gehele getal `X`; in `i` moet de bijbehorende array-index worden opgeleverd. Als zo'n getal niet bestaat moet `-1` worden geretourneerd, en moet `i` ook `-1` worden. (Voorzie alle variabelen in de functie-heading van het juiste type!)
b. (6) Schrijf een Booleaanse C++-functie `alt (A,n)` die kijkt of `A` de eigenschap heeft dat de getallen om en om stijgen en dalen, en in dat geval `true` retourneert, en anders `false`. Voor $1 \nearrow 8 \searrow 3 \nearrow 12 \searrow 5 \nearrow 11 \searrow 2$ is het dus `true`, en voor $9 \searrow 2$ ook.
c. (7) Schrijf een C++-functie `sorteer (A,n)` die het array `A` als volgt oplopend sorteert. Gebruik herhaald de functie van **a** (met geschikte `X`) en wissel het kleinste getal van het resterende deel van `A` naar de juiste plek.
d. (4) Is de sorteermethode van **c** slechter dan, even goed als, of beter dan *bubblesort*? Leg uit; in het antwoord moet `n` voorkomen.

2. (25) **a**. (6) 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. (6) Gegeven een C++-programma met daarin de volgende twee functies:

```
int ot (int r) { r += 2; return r; }//ot
int sien (int u, int v) {
  int i; y++; for ( i = 0; i <= 1000; i++ ) u += ot (v);
  v = v + 6; cout << i << "," << u << "," << v << endl; return u + v + 7;
}//sien
```

Verder zijn de globale variabelen `x` en `y` van type `int` gegeven. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = 4; y = 0; cout << sien (x,y) << ","; cout << x-1 << "," << y << endl;
```

c. (4) We voegen een `&` toe in de heading van `ot`. Wat is de uitvoer van:

```
x = 4; for ( x = 0; x < 10; x++ ) cout << ot (x) << endl;
```

d. (5) Zet nu alleen twee `&`'s in de heading van `sien`. Wat is de uitvoer van:

```
x = -2; y = 0; cout << sien (x,x) << ","; cout << x-1 << "," << y << endl;
```

e. (4) Mag in `ot` ergens `r = sien (ot (r),r);` staan? Onderscheid situaties met/zonder `&`.

3. (25) Gegeven bij deze opgave zijn een 2-dimensionaal array `int prijs[n][n];` en een 1-dimensionaal array `int reis[n]` met zekere `const int n` ≥ 2. Hierbij staat `prijs[i][j]` > 0 voor de reiskosten tussen plaatsen `i` en `j` met $0 \leq$ `i` $<$ `j` $<$ `n`. De kosten voor de terugreis zijn dus hetzelfde als die voor de heenreis. De overige array-elementen van `prijs` zijn 0. Een voorbeeld met `n = 4` staat hierboven (`prijs` links, `reis` rechts). De kosten van `1` naar `3` (en dus ook die van `3` naar `1`) zijn bijvoorbeeld 4. De reis gaat van `3` naar `1` naar `0` (en stopt bij de eerste `-1`, en anders na `n` plaatsen), met kosten $4 + 5 = 9$. Alle array-elementen in `reis` na de eerste `-1` zijn ook `-1`. Een reis als `3 1 3 -1` mag ook.

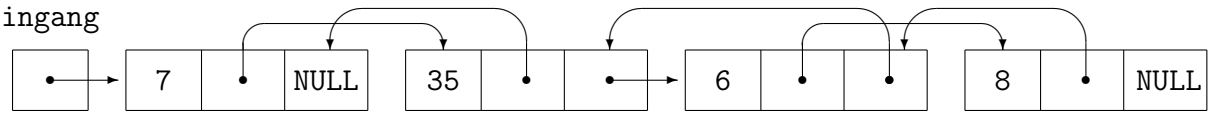
0	5	2	3	3
0	0	1	4	1
0	0	0	3	0
0	0	0	0	-1

a. (7) Schrijf een C++-functie `double gem (prijs,i)` die de gemiddelde kosten voor een directe reis vanuit `i` retourneert. Neem aan dat $0 \leq$ `i` $<$ `n`. In het voorbeeld, met `i` $= 1$: $(5 + 0 + 1 + 4)/4 = 2.5$.
b. (8) Schrijf een Booleaanse C++-functie `alle (prijs,reis,kosten)` die verifieert of de reis alle `n` plaatsen bezoekt, en daarnaast altijd in de `int` parameter `kosten` de totaalkosten van de reis oplevert.
c. (10) Schrijf een C++-functie `duur (prijs,reis,i)` die in `reis` de reis oplevert die als volgt verkregen wordt: begin in `i` en kies van daaruit de duurste directe reis (als er meerdere met dezelfde kosten zijn, kies degene met het kleinste nummer), ga daarheen, en herhaal dit tot je op een plek zou komen waar je eerder bent geweest. Tip: houd dit bij in een Booleaans array `bezocht`. De voorbeeldreis boven ontstaat zo als je in `i` $= 3$ begint.

4. (25) Gegeven is het volgende type:

```
class object { public: int info; object* volg1; object* volg2; };
```

Met behulp hiervan kan een lijst van objecten worden opgebouwd, bestaande uit vakjes met een getal, en twee pointers. Precies een van deze twee wijst naar het volgende object, de andere naar het vorige — maar je weet niet welke. Een voorbeeld, met `ingang` van type `object*`:



a. (6) Schrijf een C++-functie `voegtoe (ingang,getal)` die een nieuw object met `getal` erin netjes vooraan de lijst met ingang `ingang` toevoegt. Je mag zelf kiezen welke van de twee pointers in het nieuwe object naar vorige en volgende object wijst. Zet in het oude eerste object (als dat bestaat) de terugwijzende pointer goed.
b. (6) Schrijf een C++-functie `verwijder (ingang)` die het eerste object uit de lijst met ingang `ingang` netjes verwijdert, indien dit bestaat.
c. (4) Schrijf een C++-functie `hoogop (ingang)` die als er minstens twee objecten zijn en als het `info`-veld van het eerste object oneven is, dit ophoogt met het `info`-veld van het tweede object. In het voorbeeld: 7 wordt 42.
d. (3) In de functies bij **a**, **b** en **c** staat in de 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? Mag het, moet het? Leg duidelijk uit.
e. (6) Schrijf een C++-functie `repareer (ingang)` die ervoor zorgt dat na afloop alle `volg1`-pointers naar het volgende, en alle `volg2`-pointers naar het vorige object wijzen.