Tentamen Programmeermethoden
Woensdag 3 januari 2018
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: `www.liacs.leidenuniv.nl/~kosterswa/pm/cijf/res.html`.

1. (25) In het array `int A[n]` staan `n` (een `const` ≥ 3) gehele getallen.
a. (5) Schrijf een C++-functie `int verschil (A,n)` die het grootste verschil tussen waardes uit `A` teruggeeft. Loop daartoe precies één maal door het array.
b. (8) Neem aan dat er minstens drie verschillende waardes in `A` staan. Schrijf een C++-functie `eerste3 (A,n,a,b,c)` die de eerste drie verschillende waardes uit `A` in `a`, `b` en `c` oplevert. Zorg ervoor dat `a` $<$ `b` $<$ `c`.
c. (7) Neem aan dat `A` precies drie verschillende waardes bevat. Schrijf een C++-functie `sorteer (A,n)` die `A` als volgt oplopend sorteert. Roep één maal de functie `eerste3` van **b** aan, tel daarna hoevaak de drie gevonden getallen in `A` voorkomen, en vul tot slot `A` met de juiste waardes.
d. (5) Neem aan dat `n` een drievoud is, en verder dat `A` precies drie verschillende waardes bevat, en wel alle drie even vaak. Schrijf de C++-functie `busorteer (A,n)` die het array nu met behulp van de meest eenvoudige versie van *bubblesort* oplopend sorteert. De doorgangen = rondes moeten stoppen zodra "het derde getal aan de beurt is".

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 da (int & u, int v) {
  u = u + v; v = u - v; u = u - v;
  cout << u << "," << v << endl; return u + v;
}//da
void vinci (int a, int b) {
  int i; for ( i = 0; i < a; i++ ) { x = da (a,b); }//for
  cout << i << "," << a << "," << b << "," << x << "," << y << endl;
}//vinci
```

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 = 1; y = 3; vinci (da (x,y),y); cout << x << "," << y << endl;
```

c. (5) Idem, maar nu zonder de `&` in de heading van `da`.
d. (4) Wat levert op (maar nu weer met `&` voor de parameter `u` in `da`, zoals in **b**):

```
x = 2; y = 42; vinci (x,x); cout << x << endl;
```

e. (4) Mag ergens in `da` staan: `vinci (da (v,u),da (u,v));`? Onderscheid gevallen met en zonder `&` voor de parameters van `vinci`. Het gaat erom of de code al dan niet compileert.

3. (25) Gegeven is het twee-dimensionale array `int stemmen[m][n];`, met zekere `const int m` ≥ 2 en `const int n` ≥ 2. Er geldt dat `stemmen`[`i`][`j`] ≥ 0 het aantal stemmen van jurylid `i` op liedje `j` is. Een voorbeeld met `m = 4` en `n = 5` staat hiernaast.

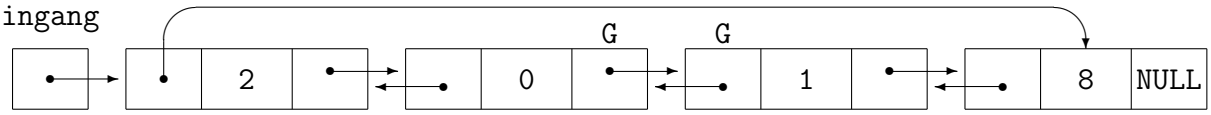
2	5	0	3	2
1	5	1	4	1
2	1	3	3	3
8	1	0	0	3

a. (8) Schrijf een Booleaanse C++-functie `eerlijk (stemmen)` die controleert of alle juryleden exact evenveel stemmen hebben uitgebracht. Oftewel: hebben alle rijen dezelfde totaalsom? In het voorbeeld: `true`.
b. (7) Twee juryleden stemmen *ongeveer hetzelfde* als voor ieder liedje geldt dat hun aantal stemmen hooguit één verschilt, en tevens dat bij minstens één liedje er evenveel stemmen zijn. Schrijf een Booleaanse C++-functie `idem (stemmen,i1,i2)` die dit controleert voor de juryleden `i1` en `i2`. Neem aan dat `0` $\leq$ `i1` $<$ `m` en `0` $<$ `i2` $<$ `m`. In het voorbeeld `true` bij `0` en `1`.
c. (10) Begin bij een zeker jurylid `i1`, met `0` $\leq$ `i1` $<$ `m`. Zoek het nieuwe jurylid `i2` dat ongeveer hetzelfde stemt als `i1` (gebruik **b**), waarbij `i2` zo klein mogelijk is. Zoek vanuit `i2` het nieuwe (verschillend van `i1` en `i2`) jurylid `i3` (`i3` zo klein mogelijk) dat ongeveer hetzelfde stemt als `i2`, enzovoorts: `i4` verschilt van `i1`, `i2` en `i3`, en stemt ongeveer hetzelfde als `i3`, ... Je stopt indien je een dergelijk nieuw jurylid niet meer kunt vinden. Schrijf een C++-functie `int vrienden (stemmen,i1)` die dit doet, en teruggeeft hoeveel juryleden je zo gevonden hebt, inclusief `i1`. Tip: gebruik een Booleaans hulparray.

4. (25) Gegeven is het volgende type:

```
class getal { public: int info; getal* vorige; getal* volgende; };
```

Met behulp hiervan kan een dubbelverbonden lijst van "getallen" worden opgebouwd, bestaande uit vakjes met een integer, en twee pointers naar respectievelijk vorige en volgende `getal`-object. Het eerste object (waar de ingangspointer naar wijst) heeft als voorganger het laatste, het laatste heeft als opvolger `NULL`. Als er slechts één getal is, is dit zijn eigen voorganger. Een voorbeeld, met `ingang` van type `getal*`:



a. (6) Schrijf een C++-functie `voegtoe (ingang,waarde)` die een nieuw `getal`-object met `waarde` erin netjes vooraan de lijst met ingang `ingang` toevoegt.
b. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste `getal`-object uit de lijst met ingang `ingang` netjes verwijdert, indien de integer in dat eerste object even is. Controleer of er minstens één `getal`-object is.
c. (5) Schrijf een C++-functie `wissel (ingang)` die de `info`-velden (*niet* de pointers) van de twee voorste `getal`-objecten omwisselt, indien er minstens drie objecten zijn. Gebruik de `vorige` pointer van het eerste om te controleren of er meer dan twee objecten zijn.
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) Neem nu aan dat de lijst minstens twee objecten heeft, en er precies één "gat" is. In het voorbeeld zou een "gat" tussen de objecten met `0` en `1` betekenen dat de pointers bij `G` beide `NULL` zijn. Schrijf een C++-functie `repareer (ingang)` die dit "gat" in de lijst repareert. Zet hier dus de `vorige` en `volgende` pointer goed.