# Tentamen Programmeermethoden
# Maandag 6 januari 2020
# 9:30–12:30 uur Informatica

Universiteit Leiden
The Netherlands

Bij alle functies moeten de variabelen (constanten eventueel 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` $\geq 3$) verschillende gehele getallen.
**a**. (6) Schrijf een Booleaanse C++-functie `gem (A,n)` die `true` geeft als er precies één array-element in `A` is (niet eerste of laatste) dat *exact* het gemiddelde is van zijn beide directe buren, en anders `false`. Dus `true` voor array `10 8 6 1`, en `false` voor `2 5 9`.
**b**. (7) Schrijf een C++-functie `int stijg (A,b,n)` die de lengte van een langste stijgende aaneengesloten deelrij van `A` geeft, en diens begin-index in `b` oplevert. Als er meerdere deelrijen het langste zijn: de kleinste `b`. Dus array `2 7 4 5 6 1 3 8` geeft 3, met `b` = 2 (deelrij `4 5 6`, even lang als `1 3 8`).
**c**. (4) We nemen in dit onderdeel aan dat `A` uit precies twee stijgende aaneengesloten deelrijen bestaat. Schrijf een C++-functie `int kl (A,n)` die het kleinste element van `A` oplevert, door de functie van **b** te gebruiken, en dan de twee kandidaten te vergelijken.
**d**. (4) Schrijf een C++-functie `bu (A,n)` die `A` *aflopend* sorteert met *bubblesort*. De functie moet stoppen als er tijdens een doorgang/ronde geen verwisselingen waren.
**e**. (4) Hoeveel vergelijkingen tussen array-elementen doet de functie van **d** minimaal en maximaal, uitgedrukt in `n`? En wanneer gebeurt dat?

**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 michael (int & a, int & b, int & c) {
  int temp = a; a = b; b = c; c = temp; return 2; }//michael
int peter (int u, int v, int w) {
  int j, a = 0; u++; for ( j = 1; j <= 1010; j++ ) a += michael (u,v,w);
  cout << j << "," << u << "," << v << "," << w << endl; return a; u++;
}//peter
```

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

```
x = 1; y = 4; z = 8; cout << peter (x,y,z) << ",";
cout << x << "," << y << "," << z << endl;
```

**c**. (4) We voegen drie `&`'s toe in de heading van `peter`, en vervangen `1010` in `peter` door `u`. Doe opnieuw **b**.
**d**. (5) We voegen weer drie `&`'s toe in de heading van `peter`, en vervangen `1010` in `peter` dit keer door `124124`. Wat gebeurt er hier:

```
x = 1; cout << peter (x,x,x) << ","; cout << x << endl;
```

**e**. (4) Compileert de code nog als in `michael` ergens `a = peter (a,a-1,a-2);` staat? Onderscheid situaties met/zonder `&`.

**3**. (25) Gegeven zijn een 2-dimensionaal array `int P[n][n];` en een 1-dimensionaal array `int K[n]` met `const int n` $\geq 2$ (bijvoorbeeld 4). Hierbij staat `P[i][j]` = `P[j][i]` $\geq 0$ voor de prijs voor een reis van `i` naar `j` ($0 \leq$ `i`, `j` $<$ `n`). In `K[i]` $> 0$ ($0 \leq$ `i` $<$ `n`) staan de kosten van een overnachting in plaats `i`.

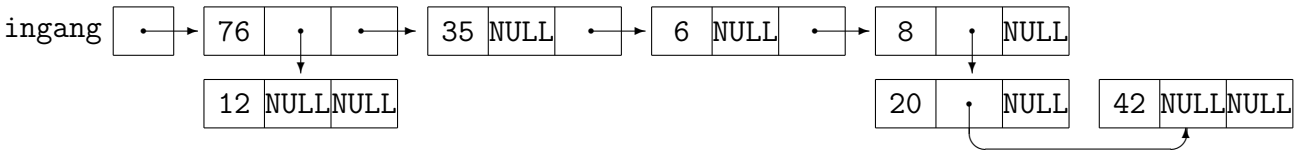
| | | | | |
|---|---|---|---|---|
| 0 | 9 | 1 | 5 | 30 |
| 9 | 0 | 2 | 5 | 10 |
| 1 | 2 | 0 | 3 | 20 |
| 5 | 5 | 3 | 0 | 15 |

**a**. (8) Schrijf een C++-functie `int trip (P,K,i,totaal)` die een goedkoopste bestemming voor een trip vanuit een gegeven `i` retourneert (als er meerdere voldoen: de kleinste index); de totaalkosten moeten in `totaal` terecht komen. Een *trip* bestaat uit heenreis (naar een andere plaats), één overnachting aldaar en terugreis. Neem aan dat $0 \leq$ `i` $<$ `n`. In het voorbeeld, met `i` $= 0$, is dat 2, waarbij `totaal` $= 1 + 20 + 1 = 22$.
**b**. (7) Schrijf een Booleaanse C++-functie `driehoek (P)` die verifieert of aan de "driehoeksongelijkheid" wordt voldaan: voor elk tweetal plaatsen moet de directe reis tussen deze twee hooguit even duur zijn als een reis via een enkele overstap (zonder overnachting).
**c**. (10) We gebruiken hier een functie `int ran ( )` die bij aanroep een random = willekeurig geheel getal tussen 0 en `INT_MAX`, grenzen inbegrepen, geeft.
Schrijf een C++-functie `int kosten (P,K,i)` die de kosten van de reis oplevert die als volgt verkregen wordt: begin in de gegeven `i` ($0 \leq$ `i` $<$ `n`), ga naar een willekeurige andere plaats, overnacht daar, en herhaal dit tot je ergens komt waar je eerder bent geweest (daar wordt niet meer overnacht). Tip: houd het bezocht zijn bij in een Booleaans array.

**4**. (25) Gegeven is het volgende type:

```
class object { public: int info; object* onder; object* rechts; };
```

Met behulp hiervan kan een lijst van objecten worden opgebouwd, bestaande uit vakjes met een getal, en twee pointers. De `rechts`-pointer wijst naar het volgende object (eventueel `NULL`), de `onder`-pointer naar een er onder gelegen object (vaak `NULL`). De `rechts`-pointers zijn alleen ongelijk `NULL` in de bovenste rij. Een voorbeeld, met `ingang` van type `object*`:




**a**. (6) Schrijf een C++-functie `erbij (ingang,getal)` die een nieuw object met `getal` erin aan de lijst met ingang `ingang` toevoegt. Als de lijst leeg was, of het eerste object al een er onder gelegen object had, moet het nieuwe object vooraan worden toegevoegd; en anders onder het eerste object.
**b**. (6) Schrijf een C++-functie `verwijder (ingang)` die het eerste object uit de lijst met ingang `ingang` netjes verwijdert, indien dit bestaat en er geen object onder zit; als er wel een object onder zit moet dit juist verwijderd worden. (In het voorbeeld: het object met 12.) Neem aan dat hier geen object meer onder zit.
**c**. (4) Schrijf een C++-functie `som (ingang)` die de waarde in het eerste object (als dat er is) ophoogt met de waardes in de objecten er onder en er rechts van (voor zover aanwezig).
**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 `print (ingang)` die de voorkomende getallen als volgt afdrukt: op regel 1 de waardes van eerste object en alle objecten daaronder, steeds voorafgegaan door een oplopende teller die op 1 begint; op regel 2 die onder het tweede object, ...: 1-76 2-12 LF 1-35 LF 1-6 LF 1-8 2-20 3-42 LF (LF is LineFeed, `'\n'`).