Tentamen Programmeermethoden
Maandag 4 januari 2021
13:00–16:00 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 Brightspace/uSis.

1. (25) In het array `int A[n]` staan `n` (een `const int` ≥ 2) verschillende gehele getallen.
a. (5) Schrijf een Booleaanse C++-functie `stijg (A,n)` die `true` geeft als het array `A` oplopend is gesorteerd.
b. (6) Schrijf een C++-functie `extreem (A,beg,eind,min,max)` die de index van het minimale element en de index van het maximale element uit het deel-array `A[beg]`...`A[eind]` geeft in respectievelijk `min` en `max`. Gebruik zo weinig mogelijk vergelijkingen tussen array-elementen. Neem aan dat `0` $\leq$ `beg` $\leq$ `eind` $<$ `n`.
c. (4) Hoeveel *vergelijkingen* tussen array-elementen doet `extreem (A,beg,eind,min,max)` minimaal en maximaal, uitgedrukt in `beg` en `eind`? Geef voor beide een voorbeeld-rijtje.
d. (4) We gebruiken de functie van **b** op geheel `A`: `extreem (A,0,n-1,min,max)`. We verwisselen dan element `A[0]` met `A[min]` en vervolgens verwisselen we element `A[n-1]` met `A[max]`. Staat het minimale element dan *altijd* in `A[0]` en het maximale element *altijd* in `A[n-1]`? Leg uit.
e. (6) Schrijf een C++-functie `sorteer (A,n)` die `A` oplopend sorteert door herhaald gebruik te maken van de functie van **b**. Denk aan **d**.

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 joe (int a, int b, int c) {
  int totaal = 0;
  for ( a = a; a < b; a++ ) totaal += c;
  cout << a << "," << totaal << endl; return totaal; }//joe
int kamala (int u, int v) {
  int i, totaal = 0;
  for ( i = 1; i <= u; i++ ) totaal += joe (u,v,i);
  cout << i << "," << u << "," << v << endl; return totaal; }//kamala
```

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 = 3; y = 5; cout << kamala (x,y) << ",";
cout << x << "," << y << endl;
```

c. (4) Geef eenvoudige formules voor de return-waardes van `joe (r,s,t)` en `kamala (r,s)`, uitgedrukt in `r`, `s` en `t`. In de formule voor `kamala` mag een sommatie blijven staan.
d. (5) Als we voor één van de vijf variabelen in de twee functie-headings een `&` toevoegen, wordt de uitvoer flink anders. Welke variabele is dat, en hoe ziet de uitvoer er dan uit?
e. (4) Compileert de code nog als in `joe` ergens `a = kamala (joe (x,y,2),a);` staat? Onderscheid situaties met/zonder `&` bij de betrokken parameters.

3. (25) Gegeven is een `m` bij `n` (beide `const int` > 0; ze hoeven bij deze opgave niet te worden doorgegeven als parameter) array `B` met hoofdletters en kleine letters. We spelen een soort *Stratego* waarbij de letters de rangen aangeven. De hoofdletters en kleine letters spelen tegen elkaar. Hiernaast een voorbeeld met `m == 4` en `n == 5`.

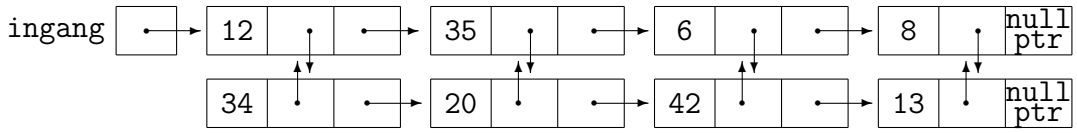
D	w	j	c	O
b	r	R	O	K
H	d	C	N	u
E	a	e	X	v

a. (7) Schrijf een C++-functie `int balans (B)` die de *balans* tussen hoofdletters en kleine letters berekent: het verschil tussen beide totaalwaardes. Hierbij tellen `A` en `a` voor 0, `B` en `b` voor 1, etc. In het voorbeeld is dit $107 - 99 = 8$ (in het voordeel van de hoofdletters).
b. (8) Nu kunnen hoofdletters horizontaal of verticaal direct aangrenzende kleine letters slaan (en vice versa) indien hun waarde echt hoger is. In het voorbeeld kan de hoofdletter `R` op positie `(1,2)` alleen kleine letter `j` op positie `(0,2)` slaan. Schrijf een Booleaanse C++-functie `slaat (B,i,j,p,q)` die controleert of er op positie `(i,j)` een hoofdletter staat die op positie `(p,q)` kan slaan. Neem aan dat $0 \leq$ `i`, `p` $<$ `m` en $0 \leq$ `j`, `q` $<$ `n`.
c. (10) Schrijf een C++-functie `int besteH (B,p,q)` die berekent welke hoofdletter de beste positie heeft. Dat betekent dat deze de aangrenzende kleine letters kan slaan met de hoogste (opgetelde) totaalwaarde. In het voorbeeld is dat `X` op positie `(3,3)`, want deze kan `e` op positie `(3,2)` en `v` op positie `(3,4)` slaan voor een totaal van $4 + 21 = 25$. Gebruik de functie van **b**. De positie van de beste hoofdletter moet worden doorgegeven via `p` en `q`, en de bijbehorende totaalwaarde wordt geretourneerd. Als er meerdere de beste zijn: één van deze. Als geen enkele hoofdletter kan slaan: geef `0` terug, ook in `p` en `q`.

4. (25) Gegeven is het volgende type:

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

Met behulp hiervan kan een structuur met twee rijen met objecten worden opgebouwd, bestaande uit vakjes met een getal, en twee pointers; zie onderstaand plaatje. Voor beide rijen geldt: de `rechts`-pointer wijst naar het volgende object er rechts naast (als laatste een `nullptr`). In de bovenste rij wijst de `onder`-pointer naar het er onder gelegen object, in de onderste rij terug naar boven. Een voorbeeld, met `ingang` van type `object*`:



a. (6) Schrijf een C++-functie `erbij (ingang,getal1,getal2)` die twee nieuwe objecten, onder elkaar, met `getal1` en `getal2` erin vooraan de lijst met ingang `ingang` toevoegt.
b. (6) Schrijf een C++-functie `verwijder (ingang)` die het eerste tweetal objecten (onder elkaar) uit de lijst met ingang `ingang` netjes verwijdert, indien deze bestaan, en beide getallen > 0 zijn. (In het voorbeeld: de objecten met 12 en 34.)
c. (4) Schrijf een C++-functie `som (ingang)` die de waarde in het eerste object (als dat er is) ophoogt met de waarde in het object er schuin rechts onder (indien aanwezig), mits het resultaat door 8 deelbaar is. In het voorbeeld: $12 \rightarrow 12 + 20 = 32$.
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 `sorteer (ingang)` die de getallen in de bovenste rij oplopend sorteert met een aangepaste *bubblesort*: wissel zolang het nodig is naburige getallen (en niet de pointers!) die verkeerd staan.