Tentamen Programmeermethoden
Woensdag 5 januari 2022
14:15–17:15 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) gehele getallen. Er komen in `A` precies twee verschillende getallen voor.
a. (7) Schrijf een Booleaanse C++-functie `oneven (A,n)` die precies dan `true` teruggeeft als er een aaneengesloten deelrijtje van `A` is dat geheel uit een oneven aantal dezelfde getallen bestaat, terwijl de eventuele getallen ervoor en erna anders zijn.
b. (6) Schrijf een C++-functie `twee (A,n,x,y,t)` die de twee verschillende getallen `x` < `y` bepaalt die in `A` voorkomen. Verder moet `t` het aantal keer worden dat `x` voorkomt.
c. (4) Schrijf een C++-functie `sorteer (A,n)` die het array `A` als volgt oplopend sorteert. Roep eerst 1× de functie van **b** aan, en zet dan de getallen direct op hun juiste plek in `A`.
d. (5) Schrijf een C++-functie `bubblesort (A,n)` die het array `A` oplopend sorteert met *bubblesort*. Stop hierbij als er een ronde/doorgang zonder verwisselingen is. (Gebruik hier nergens dat je weet dat `A` maar twee verschillende getallen bevat.)
e. (3) Hoeveel rondes doet het algoritme van **d** maximaal, uitgedrukt in de waarde van `t` die door een aanroep van de functie van **b** wordt opgeleverd? Leg uit.

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 freek (int y, int x) { int z = 1;
   while ( y > x ) { y = y - x; z++;
      cout << y << "," << z << endl; }//while
   return x * z; }//freek
int suzan (int a, int b) { int i;
   if ( a <= b ) { a = a + b; b = a - b; a = a - b; }//if
   for ( i = b; i <= a; i++ ) { a--; b += freek (a,b);
      cout << i << "," << a << "," << b << endl; }//for
   return a + b - 11; a++; }//suzan
```

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 = 8; cout << suzan (x,y) << endl; cout << x << "," << y << endl;
```

c. (4) Geef een eenvoudige formule voor de return-waarde van `freek (m,n)`, uitgedrukt in `m` en `n`, waarbij `m` > 0 en `n` > 0.
d. (5) Als **b**, maar nu met een `&` (“ampersand”) bij de parameters `y`, `x`, `a` en `b` van de functies.
e. (4) Compileert de code als in `freek` ergens `x = suzan (42,x);` staat? Onderscheid situaties met/zonder `&` bij de betrokken parameters.

3. (25) Gegeven is een `m` bij `n` (beide `const int` ≥ 2; ze hoeven bij deze opgave niet te worden doorgegeven als parameter) array `C` met gehele getallen tussen 0 en 10, grenzen inbegrepen; hierbij stelt `C[i][j]` het cijfer van het `j`-de vak van student `i` voor, en een 0 staat voor een (nog) niet gedaan vak. Hiernaast een voorbeeld met $\mathtt{m} = 4$ studenten en $\mathtt{n} = 5$ vakken.

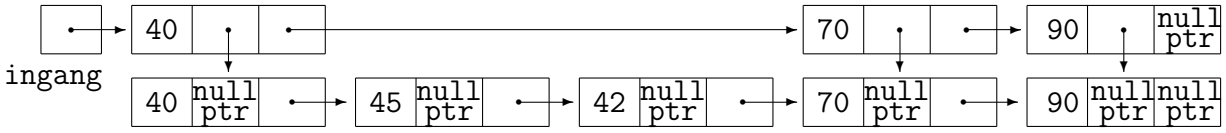
10	8	0	6	5
9	7	0	8	8
0	3	0	7	6
7	7	0	7	7

a. (7) Schrijf een C++-functie `int geen (C)` die berekent hoeveel vakken er zijn die nog door geen enkele student gedaan zijn. In het voorbeeld 1 (wegens vak 2).
b. (8) Schrijf een Booleaanse C++-functie `beter (C,i,j,tot)` die bepaalt of student `i` beter is dan student `j`. Een student is *beter* dan een andere als voor ieder vak diens cijfer echt hoger is dan dat van de andere. In `tot` moet in dat geval het totaalverschil van hun cijfers komen, en anders 0. Als beiden een vak niet hebben gedaan doet dat vak niet mee. Neem aan dat $0 \leq \mathtt{i}, \mathtt{j} < m$ en $\mathtt{i} \neq \mathtt{j}$. Voorbeeld: $\mathtt{i} = 1$, $\mathtt{j} = 2$ geeft `true`, $\mathtt{tot} = 16$.
c. (10) Schrijf een C++-functie `double debeste (C)` die bepaalt of er een student is die beter is dan *alle* anderen. In dat geval moet de gemiddelde waarde van de betreffende totaalverschillen worden teruggegeven, en anders -1. Gebruik **b**.

4. (25) Gegeven is het volgende type:

```
class vakje { public: int info; vakje* beneden; vakje* volg; };
```

Met behulp hiervan wordt een lijst met getallen (`info`) opgebouwd. Deze lijst bestaat uit twee niveaus; op het bovenste niveau staan alleen gehele tientallen; op het onderste niveau alle getallen (de tientallen staan altijd op beide niveaus). Het bovenste niveau is in die tientallen gekoppeld aan het onderste niveau met de `beneden`-pointer. De `volg`-pointer wijst steeds een volgend tiental (boven) of getal (onder) aan. De lijst begint en eindigt *altijd* met een tiental. Een voorbeeld:



a. (6) Schrijf een C++-functie `voegtoe (ingang,getal)` die twee nieuwe vakjes, onder elkaar, met `getal` erin vooraan de lijst met ingang `ingang` toevoegt indien `getal` een tiental is, en anders niets doet.
b. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste tweetal vakjes (onder elkaar) uit de lijst met ingang `ingang` netjes verwijdert, indien deze bestaan. In het voorbeeld: beide vakjes met 40. De ingang moet dan het bovenste vakje met 70 aanwijzen, het vakje met 45 is onbereikbaar! (Eigenlijk moet je eerst de functie van **e** aanroepen.)
c. (3) Schrijf een C++-functie `int tweede (ingang)` die de waarde van het tweede vakje uit de onderste rij teruggeeft, als dit bestaat, en anders -1. In het voorbeeld 45.
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. (8) Schrijf een C++-functie `ruimop (ingang)` die alle vakjes op het onderste niveau tussen het eerste en het tweede tiental (mits deze bestaan) netjes verwijdert. In het voorbeeld: de vakjes met 45 en 42.