Toetspracticum Imperatief Programmeren

4 oktober 2010

- Je kunt de opgaven in willekeurige volgorde maken.
- De eerste opgave is 1 punt waard, de overige vier opgaven elk 2 punten.
- Er wordt voor de beoordeling uitsluitend gekeken of je programma door Justitia is goedgekeurd. Er wordt dus niet naar je code gekeken, noch wordt er een verslag gevraagd.
- Als je programma voor een test in Justitia faalt, dan staat er een hint bij de test die fout gaat.
- Hopelijk overbodig om te vermelden: het is niet de bedoeling dat er wordt 'samengewerkt'. Als plagiaat geconstateerd wordt zullen zowel de kopiërende partij als de partij die gelegenheid bood tot kopiëren zich moeten verantwoorden.
- Het is toegestaan om tijdens het toetspracticum gebruik te maken van het dictaat, het boek en eventueel het internet (bijv. Nestor). Het is niet toegestaan om gebruik te maken van e-mail. Ook is het niet toegestaan gebruik te maken van laptops en mobiele telefoons.
- Bij iedere opgave is een voorbeeld invoer file `vb.in` gegeven en de bijbehorende uitvoer-file `vb.out`. Deze kun je gebruiken om zelf te testen of de uitvoer van jouw programma overeenkomt met de gevraagde layout, zodat er geen misverstand kan bestaan over spaties in de uitvoer. Voor iedere opgave kun je de voorbeeldfiles vinden bij de betreffende opgave in Justitia.
- Uiteraard mag je er vanuit gaan dat de invoer bij iedere opgave correct is. Bijvoorbeeld, als de invoer een positief getal behoort te zijn, dan hoef je niet te testen of de invoer misschien toch negatief is.

Opgave 1: Volmaakt

Een *volmaakt getal* is een positief getal dat gelijk is aan de som van al zijn delers (buiten zichzelf). Hierbij wordt 1 als deler meegerekend. Een voorbeeld van een volmaakt getal is 28, want

$$28 = 1 + 2 + 4 + 7 + 14.$$

Gevraagd wordt een programma te maken dat een positief geheel getal `n` inleest en daarna `volmaakt` afdrukt als `n` een volmaakt getal is, en `gewoon` als het getal niet volmaakt is.

Voorbeeld 1:

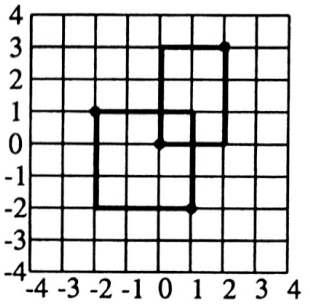
invoer:
```
28
```
uitvoer:
```
volmaakt
```

Voorbeeld 2:

invoer:
```
42
```
uitvoer:
```
gewoon
```

Opgave 2: Overlap

Hiernaast staat een raster getekend met daarin twee rechthoeken. De ene rechthoek wordt bepaald door de hoekpunten $(0, 0)$ en $(2, 3)$. De andere rechthoek wordt bepaald door de punten $(-2, 1)$ en $(1, -2)$. Hoewel het plaatje van het raster uiteraard eindig is, mag je veronderstellen dat het raster oneindig groot is. Het is duidelijk dat de twee rechthoeken overlappen. We noemen twee rechthoeken *overlappend* als ze minstens één punt gemeen hebben, dus twee rechthoeken die elkaar raken zijn ook overlappend.



Gevraagd wordt een programma te maken dat eerst de twee hoekpunten van een rechthoek inleest en daarna de twee hoekpunten van een andere rechthoek. Vervolgens drukt het programma `overlap` af als de rechthoeken overlappen en `geen overlap` als de rechthoeken niet overlappen. Je mag er vanuit gaan dat de invoer bestaat uit gehele getallen.

Voorbeeld 1 (zie figuur hierboven):

invoer:
```
0 0 2 3
-2 1 1 -2
```
uitvoer:
```
overlap
```

Voorbeeld 2:

invoer:
```
0 0 2 3
-1 0 -2 -2
```
uitvoer:
```
geen overlap
```

Opgave 3: Goldbach

In 1742 formuleerde Christian Goldbach het volgende vermoeden:

Elk even getal groter dan 2 kan geschreven worden als de som van twee priemgetallen.

Het vermoeden is tot op de dag van vandaag nog nooit door iemand bewezen. Wel is het vermoeden gecontroleerd voor alle even getallen kleiner dan 10^{18} met behulp van computers en het vermoeden blijkt voor al deze getallen te kloppen.

In deze opgave krijg je op de invoer een even natuurlijk getal `n` dat groter is dan 2 en kleiner dan 2 miljard (2×10^9). Schrijf een programma dat het kleinste priemgetal `a` en het bijbehorende priemgetal `b` bepaalt, zodanig dat de som `n` is. De uitvoer dient van de vorm `n=a+b` te zijn. Misschien ten overvloede: het getal 1 is geen priemgetal!

Voorbeeld 1:

invoer:
```
8
```
uitvoer:
```
8=3+5
```

Voorbeeld 2:

invoer:
```
876538
```
uitvoer:
```
876538=41+876497
```

Opgave 4: Verkiezingen

Na verkiezingen voor de tweede kamer moet er een winnaar bepaald worden. Uiteraard zijn de resultaten van kleine gemeenten eerder bepaald dan die van de zeer grote. Daardoor komen bij een centraal bureau de resultaten van de verschillende stembureaus na elkaar binnen en moeten daar tot een totaal worden omgezet zodat tenslotte een winnaar kan worden uitgeroepen.

In deze opgave wordt je gevraagd om een programma te schrijven dat de winnaar van de verkiezingen bepaalt, gegeven de uitslagen uit verschillende gemeentes. Op de invoer staat een positief geheel getal `n`, het aantal gemeenten. Daarna volgen er `n` uitslagen, voor iedere gemeente één. We gaan uit van 10 partijen, namelijk PVDA, CDA, VVD, PVV, GL, D66, SP, CU, PVDD en SGP. Per gemeente bestaat de uitslag uit precies 10 regels invoer. Op iedere regel staat de naam van een partij in hoofdletters gevolgd door een getal: het aantal stemmen die de partij heeft gekregen in deze gemeente.

De bedoeling is dat jouw programma de naam van de partij afdrukt die de meeste stemmen heeft verkregen en tevens het totaal aantal stemmen dat deze partij heeft gekregen. Je mag er vanuit gaan dat er altijd een winnaar is (dus een gelijk aantal stemmen komt niet voor).

Hint: Je mag gebruik maken van de onderstaande functie `indexVanString`. Deze functie zet de naam van een partij (een string) om in een getal (0 voor PVDA, 1 voor CDA, etc). De code van deze functie kun je vinden in Justitia, zodat je deze niet hoeft over te tikken.

```
int indexVanString(char str[]) {
  if (strcmp(str, "PVDA") == 0) return 0;
  if (strcmp(str, "CDA") == 0) return 1;
  if (strcmp(str, "VVD") == 0) return 2;
  if (strcmp(str, "PVV") == 0) return 3;
  if (strcmp(str, "GL") == 0) return 4;
  if (strcmp(str, "D66") == 0) return 5;
  if (strcmp(str, "SP") == 0) return 6;
  if (strcmp(str, "CU") == 0) return 7;
  if (strcmp(str, "PVDD") == 0) return 8;
  /* SGP blijft over */
  return 9;
}
```

Voorbeeld:

invoer:

```
2
VVD  1364623
PVDA 1243746
PVV  1043764
CDA   954625
GL    843567
SP    773256
D66   647832
CU    237625
SGP   158253
PVDD  100765
PVDA  473667
VVD   375647
CDA   294862
PVV   212423
GL     85554
D66    54354
SP     32443
CU     31242
PVDD   23543
SGP     4335
```

uitvoer:

```
VVD 1740270
```

Opgave 5: Vier op een rij

Vier op een rij is een spel voor twee spelers met als doel als eerste speler een aaneengesloten rij van vier schijven te vormen. Het spel wordt gespeeld op een verticaal geplaatst bord bestaand uit 7 kolommen en 6 rijen. Iedere speler beschikt over 21 schijven met een afwijkende kleur, meestal geel en rood. Het spel wordt gespeeld in beurten. De spelers laten om de beurt een schijf in één van de nog niet volle kolommen vallen. De schijf bezet altijd het laagst beschikbare vak in een kolom. Een speler wint door met vier van zijn eigen schijven een aaneengesloten rij te vormen. Een rij kan zowel verticaal, horizontaal als diagonaal worden gevormd en beindigt het spel. Het spel eindigt in *remise* als geen van de twee spelers er in slaagt een aaneengesloten rij te vormen voordat het bord volledig door schijven is gevuld. In de praktijk komt het nog wel eens voor dat het spel verder gaat terwijl reeds één van de spelers heeft gewonnen, omdat het beide spelers niet is opgevallen dat er ergens 'per ongeluk' vier-op-een-rij is gescoord. Het kan dan zelfs voorkomen dat uiteindelijk beide spelers vier-op-een-rij scoren. Zo'n geval noemen we ook *remise*. Als in een situatie geen van beide spelers vier-op-een-rij heeft gescoord, maar er zijn nog wel lege vakjes beschikbaar, dan is de stelling (nog) *onbeslist*.

In deze opgave wordt je gevraagd om een programma te schrijven die van de invoer een stelling inleest en vervolgens bepaalt of de stelling onbeslist, remise, of gewonnen is. De invoer bestaat uit 6 regels (de rijen van het bord). Op iedere regel staan 7 cijfers, voor ieder vakje één. Het cijfer 0 representeert een leeg vakje. Het cijfer 1 representeert dat de eerste speler dit vakje heeft bezet met een schijf. Uiteraard representeert het cijfer 2 dat de tweede speler het vakje heeft bezet.

Als de stelling onbeslist is, dient het programma `onbeslist` af te drukken. Bij remise moet het programma `remise` afdrukken. Bij winst moet het programma tevens bepalen wie gewonnen heeft en dat afdrukken. Bij winst van de eerste speler moet `winst 1` afgedrukt worden, terwijl bij winst voor de tweede speler de tekst `winst 2` afgedrukt moet worden.

Voorbeeld 1:

invoer:

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 2 1 1 0 0
0 1 2 2 2 1 0
0 2 1 1 2 2 0
```

uitvoer:

```
onbeslist
```

Voorbeeld 2:

invoer:

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 2 0 0 0 0 0
0 1 2 1 1 0 0
0 1 2 2 2 1 0
0 2 1 1 2 2 1
```

uitvoer:

```
winst 2
```

Voorbeeld 3:

invoer:

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 2 0 0 0 2 0
0 1 2 1 1 2 0
0 1 2 2 2 1 0
1 1 1 1 2 1 2
```

uitvoer:

```
remise
```

Voorbeeld 4:

invoer:

```
2 2 1 2 1 2 1
2 1 1 2 1 2 2
2 1 2 1 2 1 2
1 2 1 2 1 2 1
1 2 1 2 1 2 1
1 2 1 2 1 2 1
```

uitvoer:

```
remise
```