Imperatief Programmeren
TOETSPRACTICUM 2
vrijdag 21 maart 2003

Opmerking: Dit is weer een toetspracticum. Je mag gewoon je literatuur en je aantekeningen raadplegen en ook de assistenten zijn aanwezig om je te helpen.

Je stuurt je werk op voor zover je opgaven hebt kunnen afronden. Zoals gebruikelijk wordt je oplossing onderworpen aan een aantal tests. Een door het controlesysteem geaccepteerde inzending levert in ieder geval één punt. Inspectie door een van de correctoren kan deze score verhogen tot maximaal twee en een halve punt per opgave.

Aan het eind van de zitting zal er gelegenheid zijn de opgave waar je mee bezig bent ter beoordeling in te zenden, ook al is de uitwerking niet volledig. Ook zo'n incomplete uitwerking kan nog wat opleveren. Merk wel op dat we van iedereen hoogstens één onvolledige uitwerking accepteren. Deze opgave stuur je in onder de naam `NIETAF`.

Het is niet nodig de opgaven te maken in de volgorde zoals ze hieronder zijn afgedrukt. Kies die opgaven waar je het meest vertrouwen in hebt.

Typ aan het begin van de zitting het commando `toets2`. Er wordt dan bij elke opgave een directory gemaakt; bij een aantal opgaven krijg je wat hulpfiles.

Opgave 1 Kettingbrieven

Opgavenaam: `Kettingbrieven`

Heel veel mensen zullen wel eens (net als ik) hebben deelgenomen aan een kettingbrief. Hoe onschuldig ook het doel (het sturen van een prentbriefkaart, een boekje of een recept), het resultaat zal de meesten (net als mij) zwaar zijn tegengevallen.

Hoe komt dat eigenlijk? Met een beetje gereken blijkt al gauw dat het onwaarschijnlijk is dat veel deelnemers inderdaad postzakken vol reacties kunnen verwachten. Wij zullen dat rekenwerk laten uitvoeren door een programma.

We gaan er van uit dat er een slimmerik is die een kettingbrief start met het verzenden van een aantal, zeg n, brieven. Neem aan dat iedere deelnemer aan de kettingbrief ook weer n opvolgers moet vinden en dat dat ook inderdaad lukt, voorzover het **TOTALE AANTAL** deelnemers een gegeven maximum niet overschrijdt en dat daarna het spel stagneert.

Schrijf nu een programma dat bepaalt in welke 'doorzend-ronde' stagnatie optreedt. De ronde waarin de initiator zijn brieven probeert kwijt te raken nummeren we als ronde 1. Zie voor de lay out van in- en uitvoer de voorbeelden:

```
Kettingbrief
Aantal te sturen brieven:   <2>
Maximaal aantal deelnemers: <31>

Stagnatie in ronde 5
```

```
Kettingbrief
Aantal te sturen brieven:   <3>
Maximaal aantal deelnemers: <5000000>

Stagnatie in ronde 14
```

lees verder

Opgave 2 Sorteren voor de postbode

Opgavenaam: `Postbode`

In een practicumopgave hebben we een rij gehele getallen gesorteerd naar opklimmende waarde. Interpreteer deze gehele getallen nu eens als huisnummers op poststukken. Dan levert de gebruikte sorteermethode voor de postbesteller doorgaans niet de meest handige volgorde: hij/zij zal voortdurend de straat over moeten steken.

Daarom wordt nu van je gevraagd een programma te schrijven dat een rij gehele getallen zo ordent, dat eerst de oneven getallen in opklimmende volgorde komen en daarna de even getallen in dalende volgorde. (De postbode kan dan eerst de oneven huisnummers bezoeken, dan de straat oversteken en op de terugweg de even nummers behandelen.) We nemen aan, net als bij de practicumopgave, dat de huisnummers in een file staan en dat de betreffende filenaam als parameter bij de aanroep van het programma wordt meegegeven.

NB: We eisen voor de oplossing van deze opgave dat je zo veel als mogelijk gebruik maakt van zaken die je al hebt gemaakt. Je moet (helaas) wel de benodigde zaken even naar de directory van deze opgave kopieëren. Zorg er wel voor dat je niet meer dan nodig kopieert.

Voorbeeld: Bij de aangeleverde file `HuisnummerLijst` is de uitvoer:

```
3
11
11
45
62
44
34
16
```

lees verder

Opgave 3 Breuken zoeken

Opgavenaam: `Breuken`

Deze opgave gaat over niet-negatieve rationale getallen.
We vragen een programma met de volgende functionaliteit. Er zijn twee files L en M. De eerste regel van zo'n file bevat een geheel getal, het aantal regels dat volgt. Op elke volgende regel staan twee gehele getallen. Het eerste is een natuurlijk getal t; het tweede is een positief geheel getal n. Deze twee getallen representeren samen het rationale getal $\frac{t}{n}$.

De file L is geordend: als een regel met de getallen a en b komt vóór de regel met de getallen p en q, dan geldt

$$\frac{a}{b} \leq \frac{p}{q}$$

Gevraagd wordt om bij elke breuk $\frac{x}{y}$ uit M te bepalen hoeveel breuken $\frac{p}{q}$ uit L voldoen aan

$$\frac{p}{q} > (\frac{x}{y})^2$$

Voorbeeld: bij de files:

```
L:                  M:
==                  ==
13                  2
2 7                 5 3
5 17                5 9
4 3
17 8
71 31
69 29
8 3
37 13
10 3
70 21
20 6
18 5
81 15
```

levert de aanroep `java Breuken L M` de uitvoer

```
6
11
```

Schrijf een *efficiënt* programma met bovenbeschreven functionaliteit.
NB: van de file M is niet gegeven dat de regels gesorteerd zijn!

lees verder

Opgave 4 Doolhof

Opgavenaam: `Doolhof`

In deze opgave is het de bedoeling een programma te schrijven dat alle wegen tussen twee posities in een doolhof bepaalt. Een doolhof representeren we door een twee dimensionaal array. De array elementen kunnen de waarden `VRIJ` en `MUUR` aannemen. De interpretatie van deze namen zal duidelijk zijn.

Stellen we ons het programma voor als een object dat door de doolhof wandelt, dan kunnen we ons voorstellen dat het mogelijk is dat er rondjes gelopen gaan worden. Het programma zal dan nooit eindigen. Om dat te voorkomen leggen we in elk vakje dat bezocht wordt een `MUNT` neer. Naast het voorkómen van oneindige lussen kunnen deze munten ook dienen om afdrukken te maken van de gevonden routes.

Er staat een aantal files voor je klaar. Het aangeleverde programma verwacht als parameter de naam van een invoerfile. Uit deze file leest de methode `leesDoolhof` de doolhof in: een `1` wordt geïnterpreteerd als Muur en een `0` als Vrij. Symbolen anders dan `1` of `0` worden genegeerd.

Als voorbeeld invoerfile is beschikbaar de file `doolhof.inv`.

1. Ontwerp een methode die alle paden van een punt (xB, yB) naar een punt (xE, yE) bepaalt. Druk alle oplossingen af.

2. Breid je programma zo uit, dat ook de lengte van een pad wordt bepaald (en afgedrukt).

3. Breid je oplossing zo uit, dat de gebruiker het begin- en eindpunt kan invoeren. Alleen als zowel begin- als eindpunt een vrij punt van de doolhof is, kan er een geschikt pad bestaan. Voldoet de invoer hier niet aan, meld dit dan aan de gebruiker. Onze testinvoer is zo, dat begin- en eindpunt vrije punten van de doolhof zijn.

4. Breid je oplossing zo uit, dat ook de lengte van een kortste pad tussen (xB, yB) en (xE, yE) wordt afgedrukt. Is zo'n pad er niet, geef dan als uitvoer de melding:
   `Geen pad tussen opgegeven punten`

5. Verwijder nu een aantal zaken uit je programma, zodat alleen de lengte van een kortste pad wordt afgedrukt of, als er geen pad bestaat, de melding dat er geen verbinding is.

Voorbeeld: bij de gegeven voorbeeld doolhof (`doolhof.inv`)

```
Doolhof
Geef coordinaten van het beginpunt: <1 1>
Geef coordinaten van het eindpunt:  <13 14>

Lengte van een kortste pad is 57
```