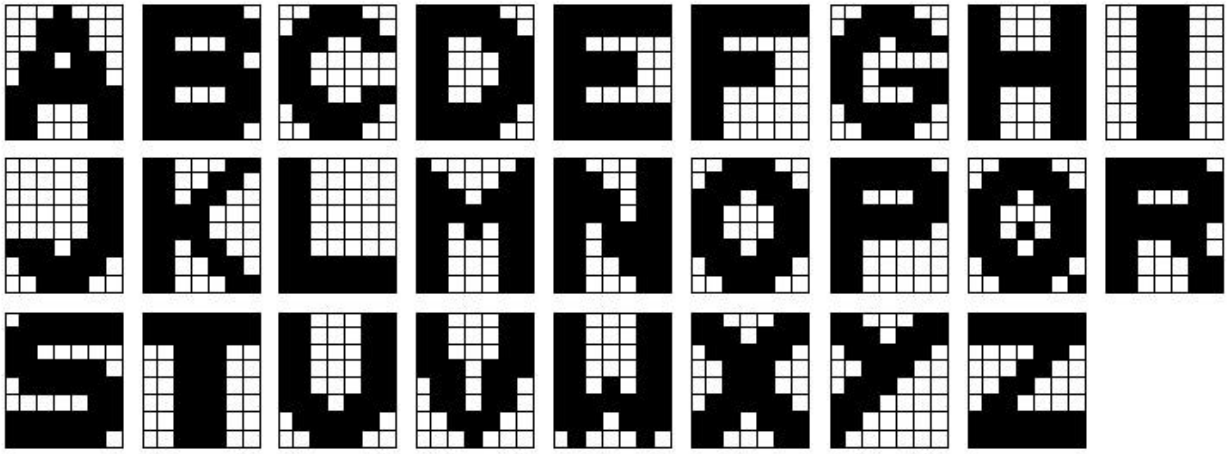
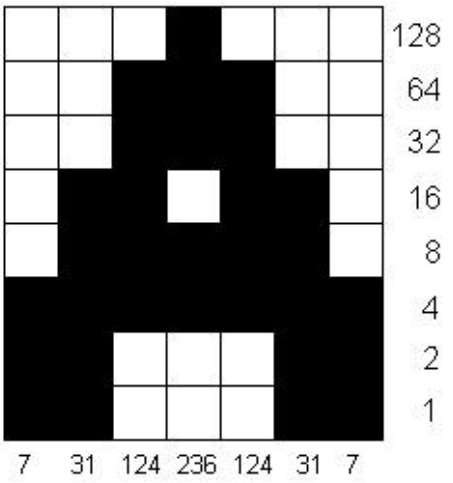
Opgave 3. Letters herkennen.

In deze opgave ga je een bescheiden begin maken met het herkennen van letters.
Om ons daarbij een beetje te beperken wordt alleen gewerkt met hoofdletters. Hieronder zie je de patronen die als uitgangspunt worden gebruikt; de gegevens uit de invoer bij de verschillende deelopgaven moeten met deze patronen worden vergeleken.



Zoals je ziet zijn deze letters vormgegeven als een serie zwarte blokjes op een veld van 8 bij 7 hokjes. Ook de patronen die je met deze "lettertekens" moet vergelijken worden gegeven als een serie van dergelijke zwarte blokjes.

De invulling van zo'n veld van 8 bij 7 hokjes kan worden weergegeven met 7 getallen. Ieder getal geeft voor een kolom van het veld aan welke hokjes zwart gekleurd zijn. Iedere rij van het veld heeft een getalswaarde (zie de figuur hieronder) en de getalswaarden van de zwarte hokjes bij elkaar opgeteld geven de getalwaarde van de kolom. De zeven getalswaarden van de kolommen (op volgorde van links naar rechts) leggen samen vast hoe het patroon eruit ziet.



Hiernaast zie je het patroon van de letter A uitvergroot weergegeven. De rijen van het veld hebben respectievelijk 128, 64, 32, 16, 8, 4, 2 en 1 als getalswaarde. Onder de zeven kolommen zie je de som van de getalswaarden van de rijen van de zwarte hokjes in de kolom.

Bestand:

Er is een bestand **`patroon.dat`** beschikbaar met een beschrijving van deze patronen. Het bestand heeft dezelfde structuur als alle invoerbestanden: De eerste regel bevat een getal N, dat aangeeft dat er N patronen in het bestand voorkomen (bij patroon.dat is N uiteraard 26). Daarna volgen N regels met elk 7 getallen k, met 0 = k = 255. Op iedere regel worden de getalswaarden van één patroon weergegeven.

De inhoud van dit bestand **`patroon.dat`** is als volgt :

```
26
7 63 124 236 124 63 7
255 255 219 219 219 255 110
60 126 231 195 195 102 36
255 255 195 195 231 126 60
```

```
255 255 219 219 219 195 195
255 255 216 216 216 192 192
60 126 227 203 235 110 44
255 255 24 24 24 255 255
0 0 255 255 255 0 0
4 6 7 3 7 254 252
255 255 24 60 102 195 129
255 255 3 3 3 3 3
255 127 48 24 48 127 255
255 255 112 60 14 255 255
60 126 231 195 231 126 60
255 216 216 216 216 216 112
60 126 199 203 199 126 61
255 216 216 216 220 223 115
115 251 219 219 219 223 206
192 192 255 255 255 192 192
252 254 7 3 7 254 252
224 252 30 3 30 252 224
254 255 6 12 6 255 254
195 231 126 60 126 231 195
199 238 124 56 112 224 192
195 199 207 219 243 227 195
```

Wanneer in één van opgaven een patroon even goed past bij twee of meer letters kies je voor de letter die het vaakst voorkomt in het Nederlands. Het bestand **`vaakst.dat`** bevat een tekstregel die bestaat uit alle 26 hoofdletters, aflopend geordend op frequentie in het Nederlands (bron: Opperlandse taal en letterkunde, Battus):

```
ENATRDOISLGHVMKUWPCBZYFJXQ
```

Dit bestand kun je zo nodig gebruiken; als een patroon net zo goed past bij een C als bij een G, dan moet je programma het beschouwen als een G, omdat de G voor de C staat in deze regel.

Invoer.

Als invoer krijgt je programma een tekstbestand **`tekst.in`**, dat dezelfde structuur heeft als het bestand **`patroon.dat`**. Op de eerste regel staat een getal *N*, met 1 = *N* = 75. Vervolgens zijn er *N* regels van elk zeven getallen; deze getallen geven een patroon aan dat jouw programma moet proberen te herkennen als een letter.

Voorbeeld: Bestand **`tekst0.in`**

```
3
224 129 129 256 129 129 224
0 62 42 42 34 54 0
0 0 28 4 4 0 0
```

Dit bestand wordt bij de voorbeelden bij alle opgaven als invoer gebruikt.

Voorbeeldbestanden en testen:

Er zijn bestanden **`tekst0.in`**, **`tekst1.in`** tot en met **`tekst5.in`** beschikbaar waarmee je je programma kunt uitproberen.
Er is een batchfile **`test3.bat`**, die je kunt gebruiken op de volgende manier:

```
test3 nio3a tekst0.in
```

Met deze opdracht test je het programma **nio3a** (of op deze plaats één van je andere programma's), waarbij vooraf eerst de invoer uit **tekst0.in** (of op deze plaats één van de andere bestanden) naar het bestand **tekst.in** wordt gekopieerd. Je zult dan zelf moeten controleren of het programma binnen de tijdlimiet stopt en de goede uitvoerfile maakt.

Opgave 3 overzicht

Onderdeel	Programma	Uitvoer	Tijdlimiet per test	Aantal testen	Punten per test	Totaal
3A	**nio3a**	**3a.uit**	2 sec	8	2	16
3B	**nio3b**	**3b.uit**	2 sec	10	2	20
3C	**nio3c**	**3c.uit**	2 sec	8	3	24
3D	**nio3d**	**3d.uit**	2 sec	10	4	40

Onderdeel 3A: Teken de invoer als plaatje

Schrijf een programma nio3a dat een bestand **tekst.in** als invoer krijgt. Het programma moet vervolgens in **3a.uit** de patronen in de invoerfile tekenen door middel van punten en X'en.
De letters komen één voor één achter elkaar, steeds gevolgd door een lege regel.

Uitvoer bij het gegeven voorbeeld: (bestand 3a.uit)

```
XXXXXXX
X..X..X
X..X..X
...X...
...X...
...X...
...X...
.XXXXX.

.......
.......
.XXXXX.
.X...X.
.XXX...
.X...X.
.XXXXX.
.......

.......
.......
.......
..X....
..X....
..XXX..
.......
.......
```

Onderdeel 3B: Tel de overeenkomsten

Schrijf een programma **`nio3b`** dat een bestand **`tekst.in`** als invoer krijgt; het programma geeft als uitvoer een tekstbestand **`3b.uit`** dat bestaat uit één regel van *N* hoofdletters. De tekst in het uitvoerbestand is de interpretatie die je programma geeft aan van de ingevoerde patronen volgens onderstaand algoritme.
Je vergelijkt ieder aangeboden patroon met de 26 letterpatronen uit **`patroon.dat`**. Daarbij tel je voor iedere letter het aantal hokjes waarvoor het patroon en het letterpatroon hetzelfde zijn. Bij een ideale overeenkomst is dat 56. Je programma kiest als interpretatie van het patroon de letter met het hoogste aantal gelijke hokjes; in geval er meerdere letters zijn waarvoor dit aantal gelijk is, wordt de letter gekozen die het eerst staat genoemd in het bestand **`vaakst.dat`**.

Uitvoer bij het gegeven voorbeeld: (bestand 3b.uit)
```
TAI
```

Enkele problemen.

Wanneer je op deze manier een patroon vergelijkt met een standaardletter kan er natuurlijk van alles misgaan. Twee voorbeelden:

- **Letterdikte.** Misschien maakt degene wiens letters herkend moeten worden ze wel wat minder vet of juist wat vetter dan de aangegeven letters.
- **Lettergrootte.** Misschien wordt in de te herkennen letters wel niet altijd het hele gebied van 8 bij 7 vakjes gebruikt.

Overgangspatronen.

Om het probleem van de lettergrootte aan te pakken beperken we ons voor elk patroon tot de rechthoek waarin zwarte vakjes te vinden zijn. In alle letters in de invoer zal altijd tenminste één vakje zwart zijn.
Om de letterdikte mee te nemen moet er gekeken worden naar kleur*overgangen*. Een manier om dit te doen is als volgt:

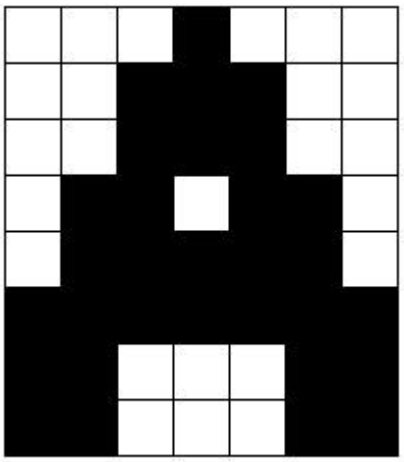
1. Schrijf het patroon door middel van W en Z op (voorbeeld 2e letter uit tekst0.in):
   ```
   ZZZZZ
   ZWWWZ
   ZZZWW
   ZWWWZ
   ZZZZZ
   ```

2. Streep nu alle dubbele Z-en en W-en weg.
   ```
   Z
   ZWZ
   ZW
   ZWZ
   Z
   ```

3. Streep tenslotte dubbele regels achter elkaar weg (komt in dit voorbeeld niet voor).

Ditzelfde proces kan herhaald worden voor de kolommen, van boven naar beneden.
Het aantal regels wat overblijft voor de rijen noemen we de rij-regels, en het aantal dat overblijft voor de kolommen de kolom-regels.

Hieronder staat nog een ander voorbeeld met de A uit **`patroon.dat`** (patroon b) en een kleiner A-patroon (patroon p):



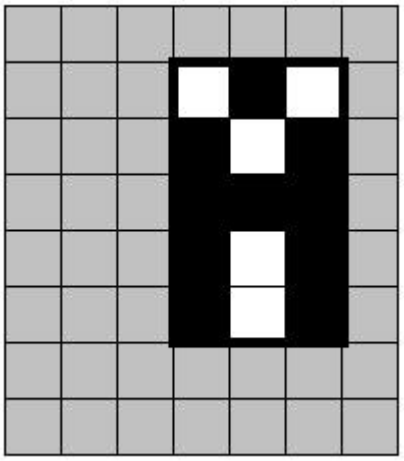
Rijen:

```
WZW
WZWZW
WZW
Z
ZWZ
```

Kolommen:

```
WZ
WZW
ZWZW
WZW
WZ
```

Aantal rij-regels R_b 5; aantal kolom-regels K_b : 5.



Rijen:

```
WZW
ZWZ
Z
ZWZ
```

Kolommen:

```
WZ
ZWZW
WZ
```

Aantal rij-regels R_p : 4; aantal kolom-regels K_p : 3.

Onderdeel 3C: Overgangspatronen bepalen

Schrijf een programma nio3d dat een bestand **tekst.in** als invoer krijgt en als uitvoer in bestand **3c.uit** de overgangspatronen van de gegeven patronen geeft.
In het onderstaande voorbeeld staat het uitvoer formaat. Merk op dat hierin de patronen corresponderend met rijen voor de kolommen komen en dat de patronen voorafgegaan worden door het aantal regels.
Na elk patroon komt een lege regel.

Uitvoer bij het gegeven voorbeeld: (bestand 3c.uit)

```
Patroon 1:
3
Z
ZWZWZ
WZW
5
ZW
ZWZ
Z
ZWZ
ZW
```

```
Patroon 2:
5
Z
ZWZ
ZW
ZWZ
Z
3
Z
ZWZWZ
ZWZ

Patroon 3:
2
ZW
Z
2
Z
WZ
```

Vergelijken door middel van overgangspatronen

Om de mate van overeenstemming tussen twee overgangspatronen te bepalen ga je als volgt te werk. Zowel voor de rijen als voor de kolommen ga je op zoek naar een zo lang mogelijke serie regels die in beide overgangspatronen voorkomen, in dezelfde volgorde. In het bovengenoemde voorbeeld zijn dat drie regels van de rijen en alle regels van de kolommen van het tweede patroon. Het aantal gemeenschappelijke rij-regels R_g en het aantal gemeenschappelijke kolom-regels K_g gebruik je in de volgende formule, die je een maat geeft voor de overeenstemming tussen de twee patronen:

$$O(b,\, p) = \frac{R_g}{R_p} + \frac{R_g}{R_b} + \frac{K_g}{K_p} + \frac{K_g}{K_b}$$

In het voorbeeld geldt dat de mate van overeenstemming gelijk is aan:

$$\frac{3}{4} + \frac{3}{5} + \frac{3}{3} + \frac{3}{5} = \frac{59}{20} = 2{,}95$$

Uiteraard kies je als interpretatie van een patroon nu die letter waarbij de mate van overeenstemming zo groot mogelijk is.

Onderdeel 3D: Vergelijken met overgangspatronen

Schrijf een programma **`nio3d`** dat een bestand **`tekst.in`** als invoer krijgt; het programma geeft als uitvoer een tekstbestand **`3d.uit`** dat bestaat uit één regel van N hoofdletters. De tekst in het uitvoerbestand is de interpretatie die je programma geeft aan van de ingevoerde patronen volgens bovenstaand algoritme op basis van de mate van overeenstemming van de overgangspatronen.

Uitvoer bij het gegeven voorbeeld: (bestand 3d.uit)
```
TEL
```