Opgave 1. Woorden en letters

In deze opgave krijgt je programma een lijst met een aantal woorden als invoer. Bij die lijst moet je bij iedere deelopgave iets uitzoeken.

je programma leest van standard input eerst een regel met een getal N (1 < N < 100). Daarna leest het N regels in, met op iedere regel precies één woord. Het woord is geschreven in hoofdletters; er komen geen bijzondere tekens voor in een woord. Elk woord bestaat uit minstens 2 en hoogstens 50 letters. Van de 26 mogelijke letters worden er maximaal 25 gebruikt in een woordenlijst.

In de helft van de testgevallen is N niet groter dan 25.

Voorbeeld:

```
18
KAARTSPEL
SCHOPPEN
HARTEN
RUITEN
KLAVEREN
AAS
HEER
VROUW
BOER
TIEN
NEGEN
ACHT
ZEVEN
ZES
VIJF
VIER
DRIE
TWEE
```

Deze invoer wordt als voorbeeld gebruikt bij de deelopgaven 1A tot en met 1G.

Bij deelopgave 1H moet je geen programma inleveren, maar een bestand.

Overzicht:

Opgave	Tijdlimiet	Testen	Per test	Totaal
1A	1	4	2	8
1B	1	4	3	12
1C	1	6	3	18
1D	1	6	4	24
1E	1	6	5	30
1F	1	6	6	36
1G	2	6	9	54
1H	nvt	1	18	18

1A: Langste woord

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma schrijft de lengte van het langste woord in de lijst naar één regel van standard output.

Uitvoer bij het voorbeeld:

```
9
```

Toelichting: Het woord `KAARTSPEL` heeft 9 letters, langere woorden komen er in de lijst niet voor.

Oplossing Isolde Hoogendoorn in C++:

```
#include <iostream>
#include <string>
using namespace std;

char a[5000];
int b;
int c;

int main(){

      cin >> b;
      b++;
      while (b > 0){
            cin.getline(a, 5000);
                  if (cin.gcount() > c){
                        c = cin.gcount() - 1;
                  }
            b--;
      }
      cout << c;
```

65 van de 68 deelnemers hadden de volledige score voor deze opgave.

1B: Frequentie beginletter

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma telt hoe vaak de eerste letter van het eerste woord voorkomt in de invoer en schrijft het aantal naar één regel van standard output.

Uitvoer bij het voorbeeld:

```
2
```

Toelichting: De eerste letter van het eerste woord is een `K`. Die komt tweemaal voor in de invoer.

Oplossing van Nick Stijger in C++:

```
#include <stdio.h>
#include <string.h>

int main(void)
{
	int i, j, n, tel;
	char woord[100][51];
	scanf("%d",&n);
	//n=aantal woorden
	for (i=0; i<n; i++)
	{
		scanf ("%s", woord[i]);
	}
	//woorden inlezen en opslaan
	tel=0;
	for (i=0; i<n; i++)
	{
		for (j=0; j<51; j++)
		{
			if (woord[0][0]==woord[i][j])
			{
				++tel;
			}
		}
	}
	printf("%d", tel);
	return 0;
}
```

58 deelnemers hadden de volledige score voor deze opgave.

1C: Ongebruikte letters

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma schrijft één regel tekst naar standard output. Op die regel staan alle letters uit het alfabet die niet gebruikt worden in de woordenlijst, op alfabetische volgorde.

Uitvoer bij het voorbeeld:

```
MQXY
```

Toelichting: De andere 22 letters komen allemaal wel voor in minstens één woord van de woordenlijst.

Oplossing van Ilia Mohammadian Moghayer in Python 2:

```
list = []
alphabet =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q'
,'R','S','T','U','V','W','X','Y','Z']
notlist = []
aantal = int(input())
alleletters = []
sum = 0
for x in range(aantal):
    woord = raw_input()
    for item in woord:
        if item not in alleletters:
            alleletters.append(item)

for x in alleletters:
    for y in alphabet:
        if x == y:
            notlist.append(x)

for x in alphabet:
    if x not in notlist:
        list.append(x)
c = ''

for x in list:
    c += x


print(c)
```

61 deelnemers hadden de volledige score voor deze opgave.

1D: Uniekeletterswoorden

Een **uniekeletterswoord** is een woord waarin iedere letter maar eenmaal voorkomt.

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma schrijft naar standard output één regel met daarop het aantal uniekeletterswoorden in de woordenlijst.

Uitvoer bij het voorbeeld:

```
10
```

Toelichting: De woorden `HARTEN`, `RUITEN`, `VROUW`, `BOER`, `TIEN`, `ACHT`, `ZES`, `VIJF`, `VIER` en `DRIE` zijn uniekeletterwoorden.

Oplossing van Marc Shair Ali in Python 3:

```
def woord(naam):
    x = []
    aantalLetters = len(naam)
    x.append(naam[0])

    nummer = 1

    while nummer != aantalLetters:
        w = int(x.count(naam[nummer]))
        if w > 0:
            return False
        if w == 0:
            x.append(naam[nummer])
            nummer = nummer+1
    return True


a = input()

aantalUnieken = 0

lijst = []
for i in range (int(a)):
    lijst.append(input())

#print(lijst)

for i in lijst:
    if woord(i) == False:
        continue
    if woord(i) == True:
        aantalUnieken = aantalUnieken +1
print(aantalUnieken)
```

57 deelnemers hadden de volledige score voor deze opgave.

1E: Stijgers en dalers

Een woord is een **stijger** als de letters van het woord op alfabetische volgorde staan.
Een woord is een **daler** als de letters van het woord als je het van achteren naar voren leest op alfabetische volgorde staan.

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma schrijft naar standard output op één regel eerste het aantal stijgers en dan het aantal dalers in de woordenlijst; de getallen worden gescheiden door een spatie.

Uitvoer bij het voorbeeld:

```
2 0
```

Toelichting: De woorden `AAS` en `ACHT` zijn stijgers. In de invoer komen geen dalers voor.

Oplossing van Jean-Paul Rozestraten in C++:

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    int N;
    int sum1 = 0;
    int sum2 = 0;
    string arr[100];
    int tmp;

    cin >> N;

    for (int i = 0; i < N; i++) {
        cin >> arr[i];
    }

    for (int i = 0; i < N; i++) {
        tmp = 1;
        for (int j = 1; j < arr[i].length(); j++) {
            if (arr[i][j] < arr[i][j - 1]) {
                tmp = 0;
                break;
            }
        }
        sum1 += tmp;
    }

    for (int i = 0; i < N; i++) {
        tmp = 1;
        for (int j = arr[i].length() - 1; j > 0; j--) {
            if (arr[i][j-1] < arr[i][j]) {
                tmp = 0;
```

```
                break;
            }
        }
        sum2 += tmp;
    }

    cout << sum1 << " " << sum2 << endl;
    return 0;
}
```

56 deelnemers hadden de volledige score voor deze opgave.

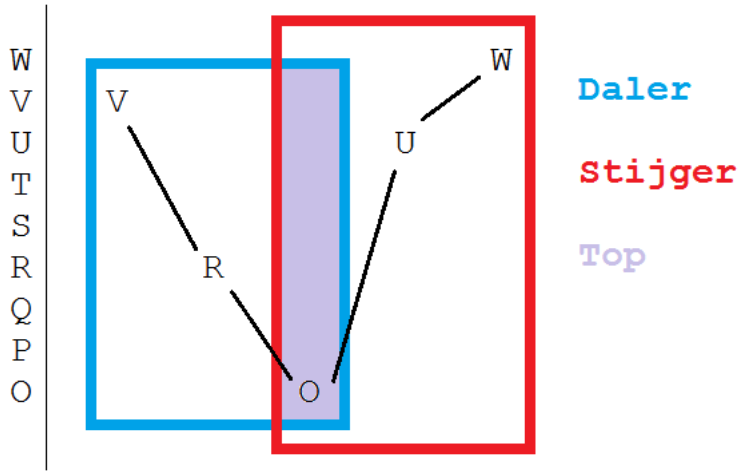
1F: Berg- en dalwoorden

Een **bergwoord** is een woord dat begint als een stijger van tenminste twee letters en eindigt als een daler van tenminste twee letters. De overlap tussen stijgerdeel en dalerdeel bestaat uit één of meer dezelfde letters; dat is de top van het woord.

Voorbeeld: `OOM` of `STOM`

Een **dalwoord** is een woord dat begint als een daler van tenminste twee letters en eindigt als een stijger van tenminste twee letters. De overlap tussen dalerdeel en stijgerdeel bestaat uit één of meer dezelfde letters; dat is de top van het woord.

Voorbeeld: `AAS` of `VROUW`



Het woord `VROUW` is een dalwoord

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma schrijft naar standard output op één regel eerst het aantal bergwoorden en dan het aantal dalwoorden in de woordenlijst; de getallen worden gescheiden door een spatie.

Uitvoer bij het voorbeeld:

```
2 6
```

Toelichting: `DRIE` en `TWEE` zijn bergwoorden, `AAS`, `HEER`, `VROUW`, `TIEN`, `ZES` en `VIER` dalwoorden.

Oplossing van Dustin Bessems in PHP:

```
<?PHP
$g = 0;

while (fscanf(STDIN, "%s\n", $String)) {
	${'l' . $g} = $String;
	$g++;
}

//Create array that saves the words put in, standard true!
for($i = 1; $i <= $g; $i++) {
	$berg[$i] = 1;
	$dal[$i] = 1;
	$stijg[$i] = 1;
	$daal[$i] = 1;
}

for($i = 1; $i <= $g; $i++) {
	//Letter Array
	$ul = array('A', 'B', 'C',
'D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T',
'U','V','W','X','Y','Z');
	$curlet = ${'l' . $i}[0];
	for($p = 0; $p < count($ul); $p++) {
		if($curlet == $ul[$p]) {
			$curind = $p;
			break;
		}
	}
	//Check voor Berg
		//Check stijgen
		for($x = 1; $x < strlen(${'l' . $i}); $x++) {
			$stijgc = false;
			for($y = 0; $y < count($ul); $y++) {
				if(${'l' . $i}[$x] == $ul[$y]) {
					if($y >= $curind) {
						$stijgc = true;
						$stijg[$i]++;
						$curind = $y;
						break;
					}
					else {
						break;
					}
				}
			}
			$next = $x;
			if($stijgc == false) {
				break;
			}
		}


		//Check Dalen
```

```
            if($stijg[$i] >= 2 && $stijg[$i] != strlen(${'l' . $i}))
{
                  for($x = $next; $x < strlen(${'l' . $i}); $x++) {
                        $daalc = false;
                        for($y = 0; $y < count($ul); $y++) {
                              if(${'l' . $i}[$x] == $ul[$y]) {
                                    if($y <= $curind) {
                                          $daalc = true;
                                          $daal[$i]++;
                                          $curind = $y;
                                          break;
                                    }
                                    else {
                                          break;
                                    }
                              }
                        }
                        if($daalc == false) {
                              $berg[$i] = 0;
                              break;
                        }
                  }
            }
            else {
                  $berg[$i] = 0;
            }

      //Check voor Dal
      if($berg[$i] != 1) {
            $stijg[$i] = 1;
            $daal[$i] = 1;
            for($p = 0; $p < count($ul); $p++) {
                  if($curlet == $ul[$p]) {
                        $curind = $p;
                        break;
                  }
            }
            //Check dalen
            for($x = 1; $x < strlen(${'l' . $i}); $x++) {
                  $daalc = false;
                  for($y = 0; $y < count($ul); $y++) {
                        if(${'l' . $i}[$x] == $ul[$y]) {
                              if($y <= $curind) {
                                    $daalc = true;
                                    $daal[$i]++;
                                    $curind = $y;
                                    break;
                              }
                              else {
                                    break;
                              }
                        }
                  }
                  $next = $x;
                  if($daalc == false) {
                        break;
```

```
                }
            }

            //Check stijgen
            if($daal[$i] >= 2 && $daal[$i] != strlen(${'l' . $i})) {
                for($x = $next; $x < strlen(${'l' . $i}); $x++) {
                    $stijgc = false;
                    for($y = 0; $y < count($ul); $y++) {
                        if(${'l' . $i}[$x] == $ul[$y]) {
                            if($y >= $curind) {
                                $stijgc = true;
                                $curind = $y;
                                break;
                            }
                            else {
                                break;
                            }
                        }
                    }
                    if($stijgc == false) {
                        $dal[$i] = 0;
                        break;
                    }
                }
            }
            else {
                $dal[$i] = 0;
            }
        }
}

$bp = 0;
$dp = 0;
for($i = 0; $i < count($berg); $i++) {
    if($berg[$i] == 1) {
        $bp++;
    }
    else if($dal[$i] == 1) {
        $dp++;
    }
}
echo $bp . ' ' . $dp;
?>
```

30 deelnemers hadden de volledige score voor deze opgave.

1G: Plakwoordenlijst

Twee woorden zijn plakwoorden als ze tenminste twee letters gemeenschappelijk hebben. HARTEN en ACHT zijn plakwoorden, net als ZEVEN en TWEE. Elke letter moet uniek worden gepaard met een letter uit het andere woord, dus ZEVEN en TWEE zijn bijvoorbeeld wel plakwoorden (door de twee E's in beide woorden), maar ZEVEN en DRIE niet.
Een plakwoordenlijst is een lijst met woorden waarvan de opvolgende twee telkens plakwoorden zijn. Een voorbeeld is TIEN, NEGEN, ZEVEN, ZES.
In een plakwoordenlijst bij de ingevoerde woordenlijst staan alle woorden in de volgorde waarin ze in de invoer voorkomen! Bij het voorbeeld is VIJF, VIER, DRIE wel een geldige plakwoordenlijst, DRIE, VIER, VIJF niet.

Schrijf een programma dat een woordenlijst inleest van standard input. Het programma zoekt de lengte van de langste plakwoordenlijst die bij de woorden in de invoer kan worden gemaakt. Je programma schrijft naar standard output één regel met daarop het aantal woorden in de langst mogelijke plakwoordenlijst.

Uitvoer bij het voorbeeld:

```
10
```

Toelichting: Er is tenminste één plakwoordenlijst van lengte 10, er is geen langere. Een voorbeeld van een plakwoordenlijst van lengte 10 is de volgende:

```
KAARTSPEL
SCHOPPEN
HARTEN
RUITEN
KLAVEREN
TIEN
NEGEN
ZEVEN
VIER
DRIE
```

Oplossing van Robert Koprinkov in C++

```
#include <iostream>
#include <cstring>

using namespace std;

bool isPlak(string s1, string s2){
      int m = 0;
      for(int n=0; n<s1.length(); n++){
            for(int j=0; j<s2.length(); j++){
                  if(s1[n]==s2[j]){
                        m++;
```

```
                        s1[n] = s2[j] = 'a'; //to letter nothing else
match so no use again
                  }
            }
      }
      if(m>=2){
            return true;
      } else {
            return false;
      }
}

int main(){
      int N;
      cin>>N;
      string w[110];
      int v[110];
      memset(v, 0, sizeof(v));
      for(int n=0; n<N; n++){
            cin>>w[n];
            v[n] = 1;
      }
      //cout << isPlak("KAARTSPEL", "SCHOPPEN") << endl;
      int maxv = 0;
      //cout << N << endl;
      for(int n=0; n<N; n++){
      //cout << n << endl;
            for(int j=0; j<n; j++){
      //          cout << w[n] << " " << w[j] << endl;
                  if(isPlak(w[n], w[j])){
      //                cout << w[n] << endl;
      //                cout << w[j] << endl;
                        if(v[n]<v[j]+1){
                              v[n] = v[j]+1;
                              maxv = max(maxv, v[n]);
                        }
                  }
            }
      }
      cout << maxv << endl;
      return 0;
}
```

Alleen Christel van Diepen, Arend Mellendijk en Robert Koprinkov hadden de volledige score voor deze opgave.

De techniek om deze opgave efficiënt op te lossen heet dynamisch programmeren. Zie voor een uitleg bijvoorbeeld http://liacs.leidenuniv.nl/~vlietrvan1/hslava/dynamischprogrammeren.pdf

1H: Woordenketting

Een **woordenketting** is een rij woorden waarbij ieder woord begint met de laatste letter van het vorige woord. De laatste letter van het laatste woord gelijk is aan de eerste letter van het eerste woord.

Bij deze deelopgave ga je op zoek naar een bijzondere woordenketting in een lange woordenlijst. De woordenketting mag namelijk iedere letter maximaal eenmaal als begin- en eindletter hebben.

De woordenlijst is ditmaal beschikbaar voor je in de vorm van een bestand `words.dat`, dat als bijlage is meegegeven in het wedstrijdsysteem. Deze woordenlijst bestaat uit 94235 woorden van minstens twee en hoogstens twaalf hoofdletters, met op iedere regel één woord. Het bestand wordt afgesloten met een regel met het teken "#". Verder bevat het bestand geen speciale tekens. Let op dat dit dus een ander formaat dan in de eerdere deelopgaven: op de eerste regel wordt NIET het aantal woorden gegeven; om de laatste regel te vinden wordt de lijst afgesloten met een speciaal teken.

Voor deze opgave lever je geen programma in, alleen de beste oplossing die je op wat voor manier dan ook gevonden hebt.

Jouw oplossing stuur je in in het wedstrijdsysteem als tekstbestand. Het is een tekstbestand met op de eerste regel een getal S, dat zegt uit hoeveel woorden je ketting bestaat, gevolgd door S regels met op iedere regel één woord.

Voorbeeld:

```
4
AAH
HAAF
FABLE
EYRA
```

Voor een oplossing van meer dan vier woorden krijg je punten; hoe langer de ketting, des te meer punten je kunt verdienen.

Er waren drie deelnemers met een langst mogelijke woordenketting van 25:

Jelmer Hinssen	Wietze Koops	Sven Holtrop
25	25	25
ABSORB	ABSORB	ABSORB
BACCHIC	BICONVEX	BACCHIC
CABALLED	XI	CABALLED
DABBLE	INLAY	DAFF
EARMUFF	YTTRIC	FABLING
FABLING	CALCIFIED	GALAH
GALAH	DEV	HADJI
HADJ	VALKYRIE	ICEBLINK
JINNI	EARMUFF	KAIL
ICEBLINK	FABLING	LABARUM
KAIL	GUNSMITH	MACAROON
LABARUM	HAJ	NARCO
MACAROON	JACK	OILCAMP
NARCO	KRAAL	PABULAR
OILCAMP	LABDANUM	RAJ
PABULAR	MACAROON	JABBERERS
RABATOS	NARCO	SABBAT
SABBAT	OUTCROP	TAV
TAV	PALER	VAU
VAU	RARENESSES	UNDERJAW
UNDERJAW	SCIOLIST	WAX
WALTZ	TABLEAU	XENOGAMY
ZAX	UNDERJAW	YARDAGE
XENOGAMY	WHIZZ	ERSATZ
YEA	ZOOMANIA	ZAMARRA