Adventskalender 2014

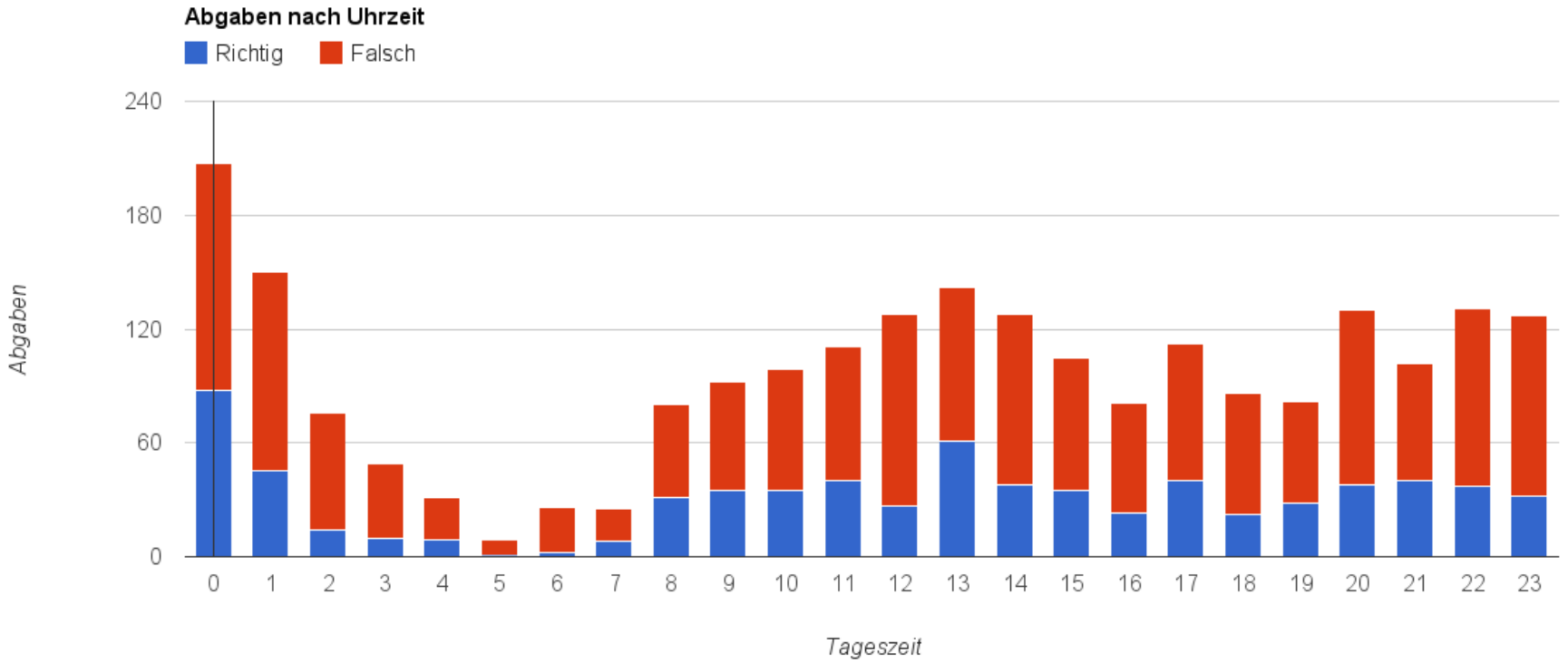
Rückblick und Preisverleihung

Zahlen und Fakten

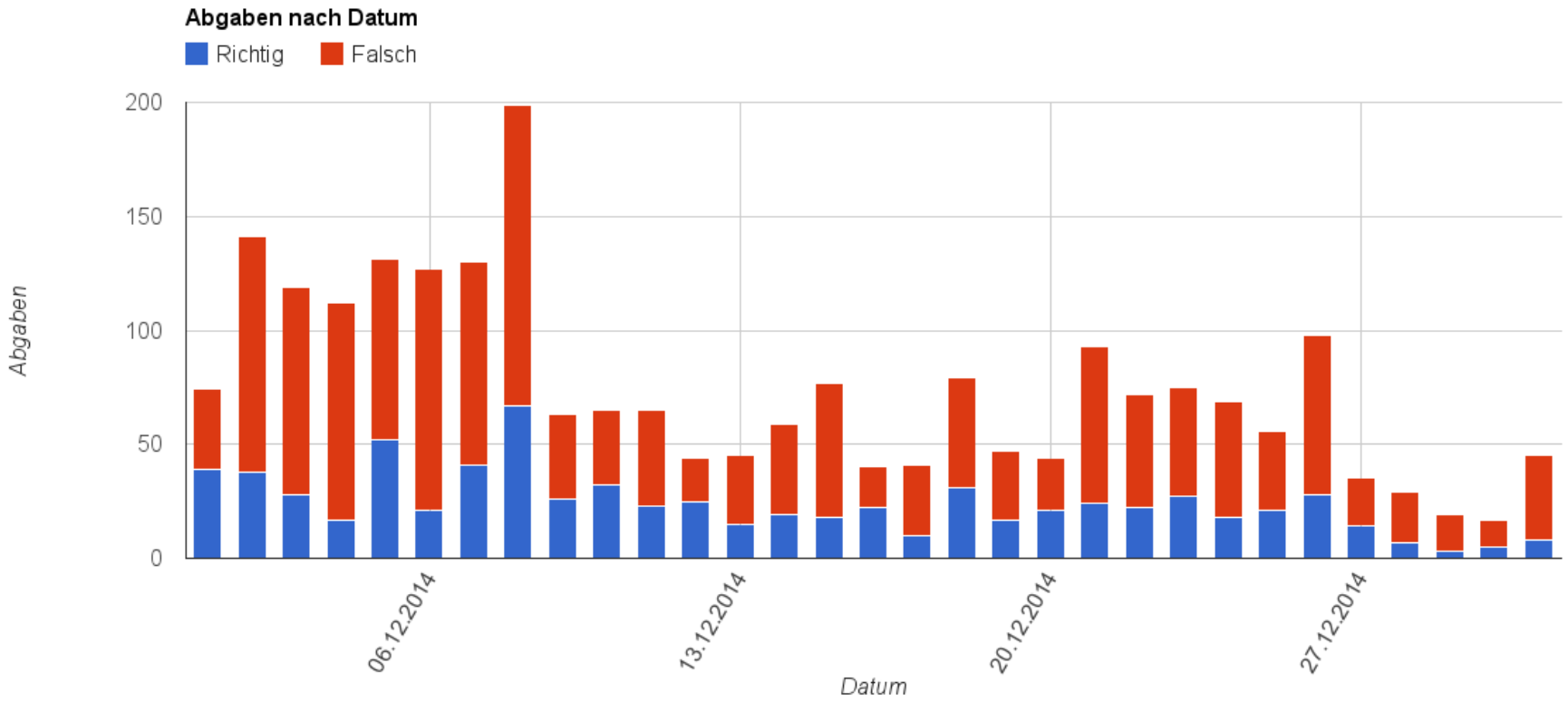
- 110 registrierte Benutzer
  - Davon 91 mit mindestens einer Abgabe
- 2309 Abgaben
  - 748 richtige
  - 1570 fehlerhafte
- 95 Testfälle
  - 8505 Testläufe
- 23 gelöste Aufgaben

- 1716 “Besucher”
- 262985 Requests
- Betriebsysteme
  - Linux (51%)
  - Windows (41%)
- Browser
  - Firefox (46%)
  - Chrome (43%)
- 1. Treffer für:
  - “wirkradius schneekanone”

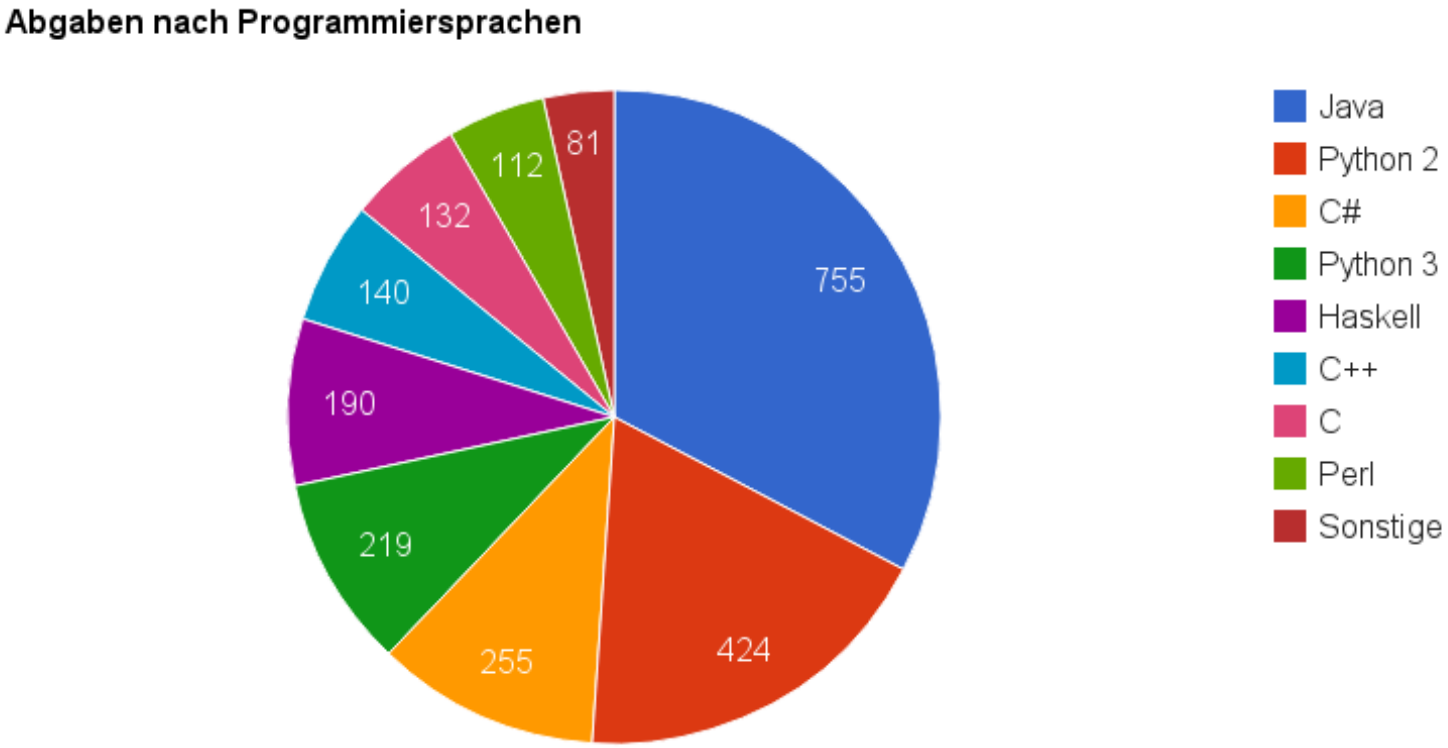
Abgaben nach Tageszeit



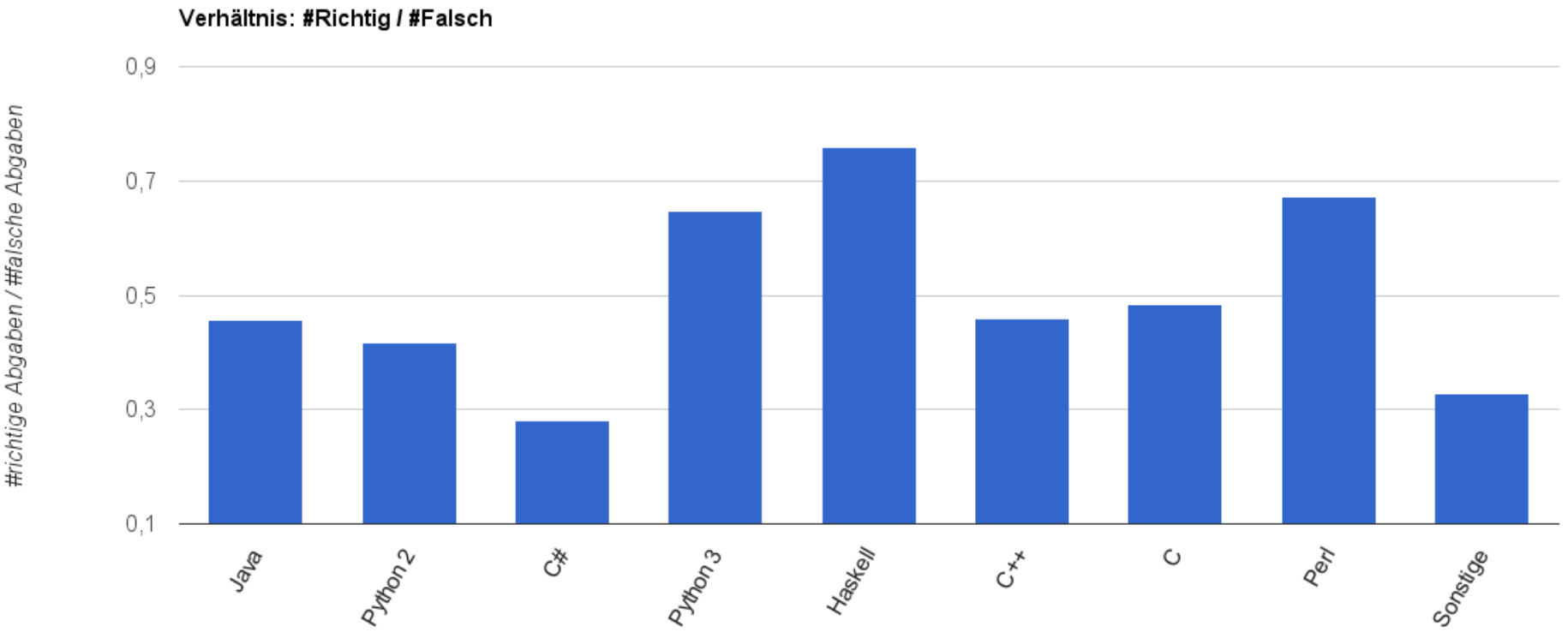
Abgaben nach Datum



Programmiersprachen



Programmiersprachen



`adventcalender.operate()`

Highlights:

- Keine Downtimes, Probleme oder Breaches
- dennoch kritische Lücke: CVE-2014-9090
- schnelle Abarbeitung der Tests: < 2 min

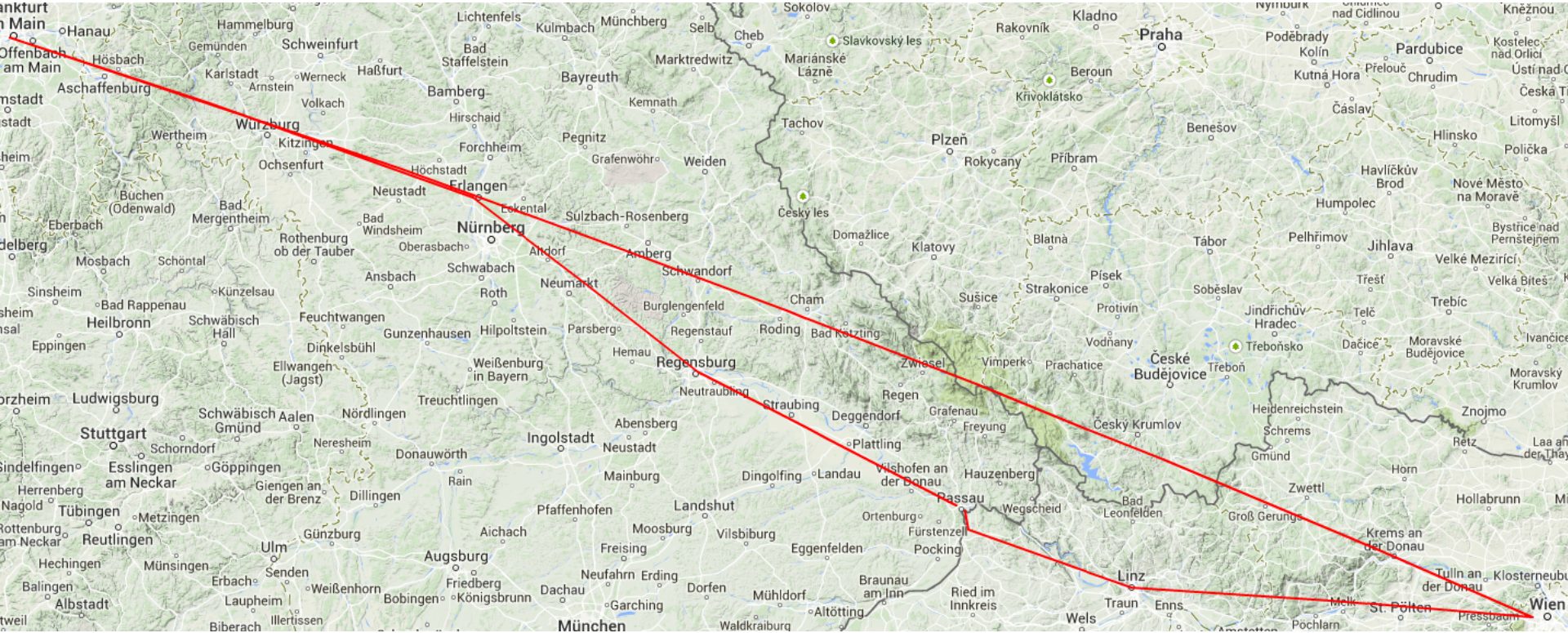
adventcalender.operate(BACKEND)

adv


wow
so enterprisy
Gentoo Linux
so scalable
very safe
SELinux
much protection
Polyinstantiation
much chroot
many languages
OpenVPN
so relocatable
Akka
much PaX

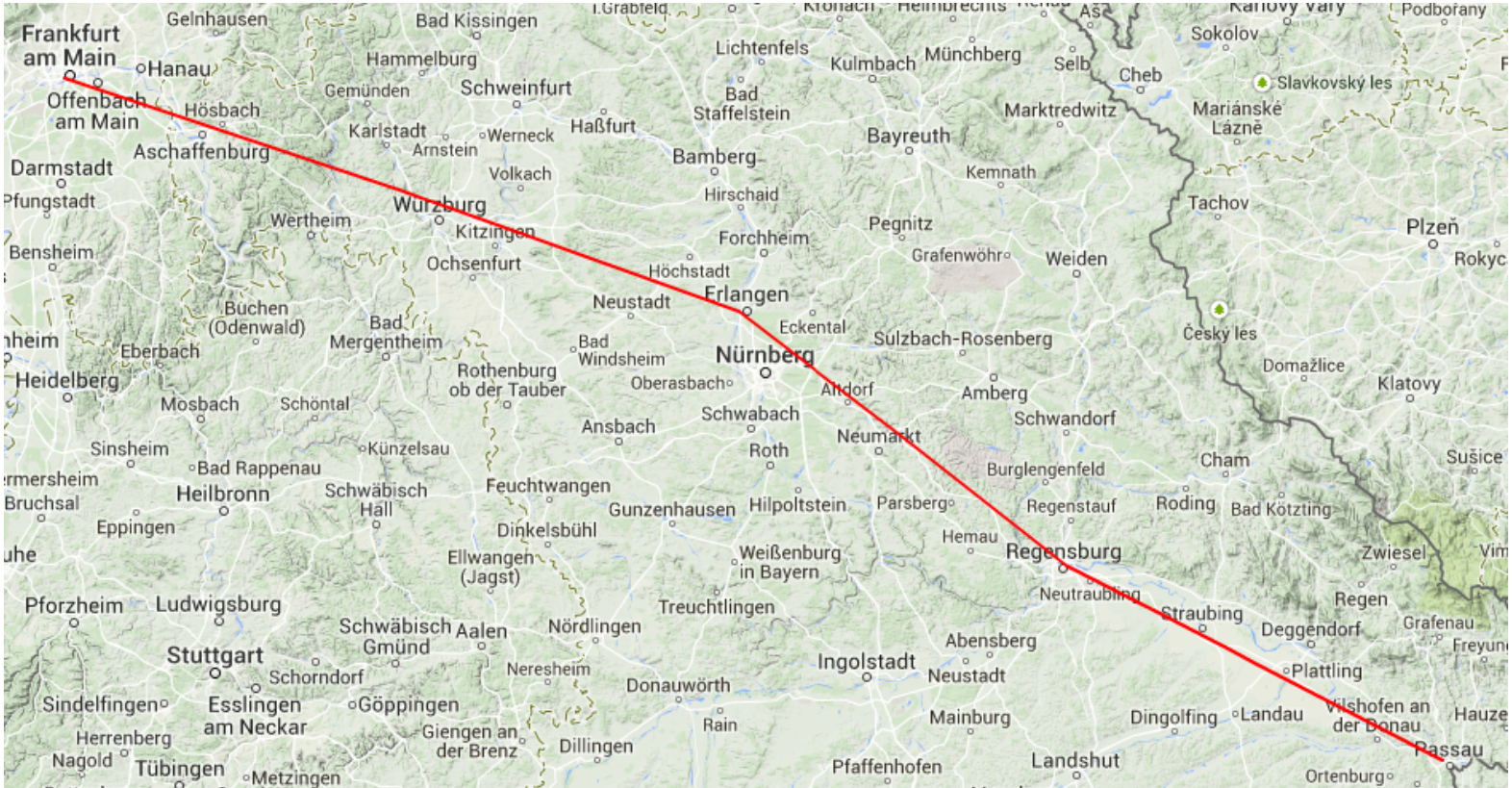
Backend Housing (erster Versuch)

Backend Housing (erster Versuch)



Backend Housing (Serverraum Uni)

Backend Housing (Serverraum Uni)



Karte © Google

// WTF does this do?

```
65    protected String minutes2String()
66    {
67        switch(minutes)
68        {
69        case 30:
70            return "HALF";
71        case 15:
72        case 45:
73            return "QUARTER";
74        case 20:
75        case 40:
76            return "TWENTY MINUTES";
77        case 10:
78        case 50:
79            return "TEN MINUTES";
80        case 5:
81        case 55:
82            return "FIVE MINUTES";
83        case 25:
84        case 35:
85            return "TWENTY FIVE MINUTES";
86        default:
87            return "WTF MINUTES";
88        }
89    }
```

```
14  my %koatn = (   cc 'Auszogne' => 437,
15          cc 'Backhendl' => 322,
16          cc 'Brennsuppe' => 912,
17          cc 'Brotzeitteller' => 123,
18          cc 'Currywurst' => 233,
19          cc 'Currywurscht' => 233,
```

```
526 void free_pinguin(Pinguin *p)
527 {
528     free_vector_relation(&(p->from));
529     //free_vector_relation(&(p->to));
530     free(p);
531 }
```

```
23  for ( int j = 0; j < mampf.length;j++) {
24      if ( !futter.containsKey(mampf[j]) )
25      if ( futter.containsKey("u1") ){
26          futter.put( mampf[j], (int) futter.get("u1") );
27          futter.remove("u1");
28      } else if ( futter.containsKey("u2")) {
29          futter.put( mampf[j], (int) futter.get("u2") );
30          futter.remove("u2");
31      } else  System.out.println("BOCKMIST");
```

```
24  if (!int.TryParse(input, out n) || n < 1 || n > 1000) {
25      // Reject bullshit
26      Console.Error.WriteLine(
```

BeerFactory.createBeer()

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. BRAUEREI.
000300
000400 DATA DIVISION.
000500 WORKING-STORAGE SECTION.
000600 01  NTestcases          PIC 9(10).
000700 01  Testcase            PIC 9(10).
000800 01  TMP                 PIC 9(10).
000900 01  Palettes            PIC ZZZZZZZZZ9.
001000 01  Boxes               PIC ZZZZZZZZZ9.
001100 01  Bottles             PIC ZZZZZZZZZ9.
001200
001300 PROCEDURE DIVISION.
001400  A-PARA.
001500  ACCEPT NTestcases
001600  PERFORM B-PARA NTestcases TIMES.
001700  STOP RUN.
001800
001900  B-PARA.
002000  ACCEPT Testcase.
002100  DIVIDE Testcase BY 450 GIVING Palettes REMAINDER Testcase.
002200  DIVIDE Testcase BY 10 GIVING Boxes REMAINDER Testcase.
002300  MULTIPLY Testcase BY 2 GIVING Bottles.
002400  DISPLAY "p="FUNCTION TRIM(Palettes)" k="FUNCTION TRIM(Boxes)
002500 " f="FUNCTION TRIM(Bottles).
```

```
#Brauerei
a = int(input())

p=0
k=0
f=0
for i in range(a):
    l=int(input())

    p=l // 450
    l -= p*450
    k=l // 10
    l -= k*10
    f=l * 2

    print("p=",p,"k=",k,"f=",f)
```

Doppelt hält besser

```
10 static void Main(string[] args)
11 {
12   int cases = int.Parse(Console.ReadLine());
13   Random rnd = new Random();
14   for (int i = 0; i < cases; i++)
15   {
16     string tmp = Console.ReadLine();
17     string[] split = tmp.Split(' ');
18     if (rnd.Next(2) == 0)
19       Console.WriteLine(V1(split));
20     else
21       Console.WriteLine(V2(split));
22   }
23   //Console.ReadKey();
24 }
25
26 public static int V1(string[] split)
27 {
28   ////////////// boring algorithm //////////////////
29   // [...]
30 }
31
32 public static int V2(string[] split)
33 {
34   ////////////// interesting algorithm //////////////
35   // [...]
36 }
37
```

Codegolf

```
nTestcases = input()

for _ in range(nTestcases):
  n = input()
  print (n * (n - 1) * (n - 2) * (n**3 + 18 * n ** 2 - 43 * n + 60) / 720)
```

```
num = int(raw_input())
for i in xrange(num):
    line = raw_input()
    Ws,Hs = zip(*[eval(x) for x in line.split()])
    print(min(Ws)*min(Hs))
```

```
import Control.Monad

main = readLn >>= void . flip replicateM (readLn >>= print . f)
f n = n*(n-1)*(n-2)*(n^3 + 18*n^2 - 43*n + 60) `div` 720
```

```
#!/usr/bin/perl
$n=<>;while($n--){my(%m,%e);$m{$_}++for split"",~~<>;$e{$m{$_}%2}++for keys%
m;print$e{1}<3?"JA\n":"NEIN\n";}
```

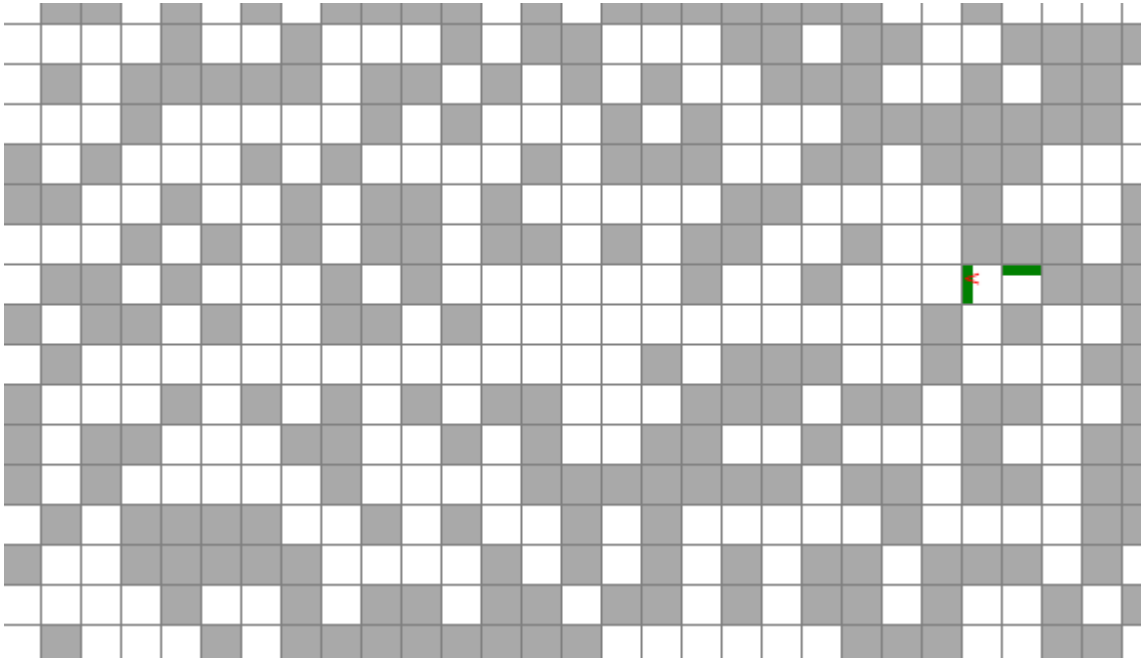
```
#!/usr/bin/perl
$n=<>;while($n--){$r=0;$r=($r+1)/$_ for split " ",~~<>;print int(0.5+$r)."\n
";}
```

Timeout Driven Development

```
35      static boolean solveable(ArrayList<Bag> bags, ArrayList<Present> present
    s, long startTime) {
36
37          if (presents.isEmpty()) {
38              return true;
39          } else if (System.currentTimeMillis() - startTime > 1200) {
40              return false;
41              /*
42               Heuristik: Durch geschicktes Sortieren der Geschenke und Saecke
     wird eine Loesung mit hoher Wahrscheinlichkeit sehr frueh gefunden.
43               Es muessen maximal 50 Faelle in 60 Sekunden bearbeitet werden,
    ein Fall sollte also nicht laenger als 1,2 Sekunden dauern.
44               Falls bis dahin kein Ergebnis gefunden wurde, ist die Wahrschei
    nlichkeit nahezu 0%, dass das Problem loesbar ist, also wird false ausgegebe
    n.
45               */
46          }
47
```

```
39      starttime = time()
40      elapsed = 0
41      while elapsed < timeout:
42          if try_pack(bins, items, rng):
43              return True
44          elapsed = time() - starttime
45
46      return False
47
```

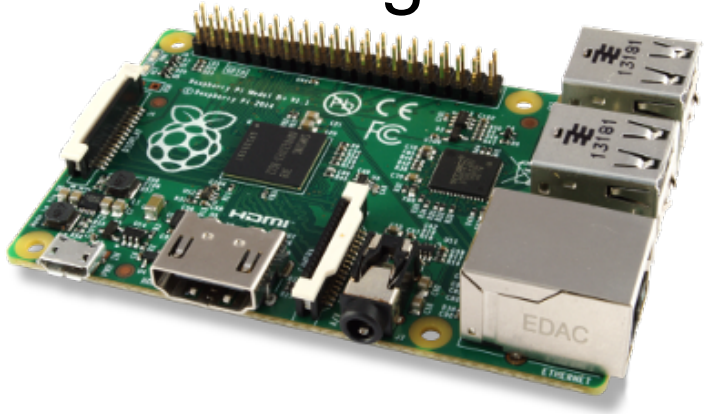
Abgaben mit GUI



Dynamischer Weihnachtsbaum

Preise

- Platz 1: Raspberry Pi Ultimate Starter Kit
- Platz 2: 50 € Heise Abo Gutschein
- Platz 3: 30 € Saturn Gutschein
- Platz 4-6: Weihnachtliche Überraschung



Die Besten: 2315 Punkte erreicht

- 3_of_8
- alefu
- ChiliconCarne
- conan_the_librarian
- Frodewin
- Mat2095

Die Platzierung der Gewinner

1. alefu
2. 3_of_8
3. conan_the_librarian
4. Frodewin
5. ChiliconCarne
6. Mat2095

Vielen Dank an alle HelfX!

debian POWERED Frontend	Lst. Lengauer Aufgaben	Marco Z. Hardware	ACTLab Aufgaben	Julian B. Hardware	Florian S. Jury	grsecurity Backend	Thomas S. Aufgaben
Philipp W. Hardware	UNIVERSITÄT PASSAU Hardware	Philip H. Aufgaben	Alexander G. Jury	PostgreSQL Powered Datenbank	Alexander B. Backend	Play! Framework	Mathias D. Jury
NGINX Frontend	Max G. PR	FORWISS Aufgaben	entoo powered Backend	Janet S. Jury	Stefan G. Jury	vmware Virtualisierung	Norbert S. Jury
Alexander B. Operations	Armin G. Aufgaben	Scala Frontend	Mathias D. Aufgaben	IEEE Hosting	Philip H. PR	Olaf L. Jury	SELinux Backend

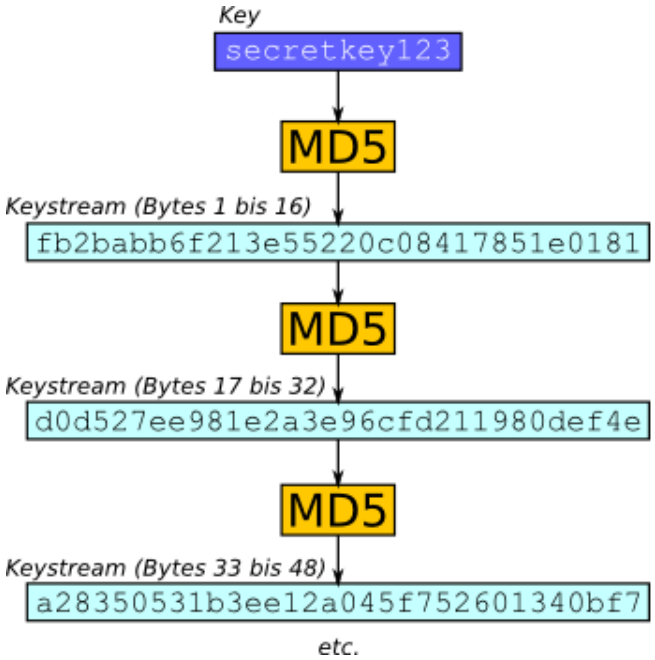
Wie sieht es denn 2015 aus?

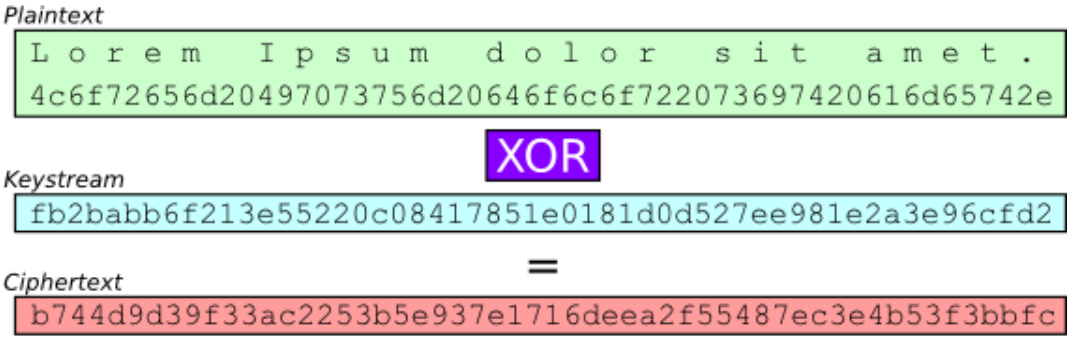
Es wird ein Team von Menschen benötigt, die

- neue Aufgaben finden
- Referenzimplementierungen von Aufgaben erstellen
- Testfälle für die neuen Aufgaben schreiben
- eine nette Einführung zu der Aufgabe erschaffen
- die Aufgabe testen und Implementieren
- das Frontend pflegen
- sich um den Support kümmern
- das Frontend weiterentwickeln
- während des Kalenders sich um Operation kümmern
- das Backend erstellen
- die Sprachen in das Backend einpflegen
- sicherstellen, dass die Auswertung funktioniert
- sich um Preise kümmern
- bei Juroren anfragen
- weitere Bewertungmetriken überlegen
- neue Ideen mit einbringen
- Interesse an dem Projekt haben
- Ressourcen haben, das Projekt fortzuführen

tl;dr: viel Zeit und Engagement für das Projekt mitbringen

MD5 Stream Cipher





MD5 Stream Cipher: Testfall 1

```
52 56 C1 F4 CD F5 D1 41 38 46 4D 17 13 AD 4A 9D 57 3D 0D 98 8E A7 14 88 18
FC 1E 81 B0 AC 11 8C E8 C0 26 16 D0 31 A4 6C 15 05 24 AD FC 09 8A B8 63 DF
42 03 52 F8 30 77 1C D5 A7 CC 8C EE 4B 7E C9 03 18 36 D6 17 64 51 93 B7 EA
1C 33 3D AE D9 96 CB 67 5D 10 D3 62 DF B5 2F 67 64 C0 93 B5 5D 6A 94 FC 90
54 93 35 EA 3A 38 56 3B 15 01 70 B6 29 86 30 D7 7B BD FD 7C 99 63 60 14 DC
02 CC DB A5 73 E3 A5 EA 78 F0 F4 1C 6A 08 86 F7 79 9C 29 37 80 48 4B 1E 49
68 6D 17 C7 88 C8 F6 8B 16 A7 B5 FC D2 43 0B B3 0A 46 AD DA D0 33 57 5A 09
EB F5 7F 60 22 25 02 26 33 9F DA AF 53 05 27 92 31 CA 1E B8 70 8E FA 09 BB
10 E8 00 2B EF 5A 06 97 35 81 63 70 90 55 31 EA 2D 60 72 08 0D 0B 6B 68 EC
42 2B 20 CE 4F 32 EC C1 FB 16 57 14 C9 A4 55 ED 0C 2B 14 28 BE CE 24 37 0D
B9 CB 85 24 A6 E5 53 65 91 C4 0D 57 D3 C7 5A 3E CD 06 2A D9 A9 8F AE BA FE
DA BB 09 DB 06 EE EC 14 16 74 B6 B1 F1 AA CA 5D B6 99 4F 41 04 A7 E9 48 2B
D7 12 2E FE 2D 49 6A C8 47 EB 77 4F 0B FD 35 6A 32 CE B3 97 79 74 69 76 4E
62 F8 12 B2 6B E9 2B 53 D5 47 E5 46 F6 C6 7D C8 18 E5 96 33 D6 6F 96 1C 2C
57 BB CC 9A 4E 5D 91 8C 6A 99 7F 3D F2 74 BD 3D 34 A3 0A 90 98 BD 4E A1 45
E9 1B A3 65 37 F6 4C 56 82
```

MD5 Stream Cipher: Testfall 1

Ciphertext

5256c1f4cdf5d14138464d1713ad4a9d
573d0d988ea7148818fc1e81b0ac118c
e8c02616d031a46c150524adfc098ab8
63df420352f830771cd5a7cc8cee4b7e
c9031836d617645193b7ea1c333daed9
[...]
ed0c2b1428bece24370db9cb8524a6e5
536591c40d57d3c75a3ecd062ad9a98f
aebafedabb09db06eeec141674b6b1f1
aaca5db6994f4104a7e9482bd7122efe
2d496ac847eb774f0bfd356a32ceb397
797469764e62f812b26be92b53d547e5
46f6c67dc818e59633d66f961c2c57bb
cc9a4e5d918c6a997f3df274bd3d34a3
0a9098bd4ea145e91ba36537f64c5682

md5(e8c02616d031a46c150524adfc098ab8)
= 63df420352f830771cd5a7cc8cee4b7e

…

md5(536591c40d57d3c75a3ecd062ad9a98f)
= aebafedabb09db06eeec141674b6b1f1

MD5 Stream Cipher: Testfall 1

```
Ciphertext                              Keystream                              Plaintext
5256c1f4cdf5d14138464d1713ad4a9d xor                                        =
573d0d988ea7148818fc1e81b0ac118c xor                                        =
e8c02616d031a46c150524adfc098ab8 xor                                        =
63df420352f830771cd5a7cc8cee4b7e xor 63df420352f830771cd5a7cc8cee4b7e = 00000000000000000000000000000000
c9031836d617645193b7ea1c333daed9 xor c9031836d617645193b7ea1c333daed9 = 00000000000000000000000000000000
[...]
536591c40d57d3c75a3ecd062ad9a98f xor 536591c40d57d3c75a3ecd062ad9a98f = 00000000000000000000000000000000
aebafedabb09db06eeec141674b6b1f1 xor aebafedabb09db06eeec141674b6b1f1 = 00000000000000000000000000000000
aaca5db6994f4104a7e9482bd7122efe xor 3d56daf7d1f7cd2a969da54316356c51 = 979c874148b88c2e3174ed68c12742af
2d496ac847eb774f0bfd356a32ceb397 xor db26458a7d1f79dce407dbe0090dc7e0 = f66f2f423af40e93effaee8a3bc37477
797469764e62f812b26be92b53d547e5 xor 45190c053d039f778c238c473fba67b2 = 3c6d6573736167653e48656c6c6f2057
46f6c67dc818e59633d66f961c2c57bb xor 2984aa19e924cafb56a51cf77b4969b1 = 6f726c64213c2f6d6573736167653e0a
cc9a4e5d918c6a997f3df274bd3d34a3 xor 621ad1b5ba2eeeb2ffeba309675e051b = ae809fe82ba2842b80d6517dda6331b8
0a9098bd4ea145e91ba36537f64c5682 xor 487f3e9c5b3770754c38382e5680804a = 42efa6211596359c579b5d19a0ccd6c8
```

MD5 Stream Cipher: Testfall 3

```
Ciphertext                          Keystream                          Plaintext
[...]
1baaacce2a0e80094f0b617e3f4d09ad xor                                =
c03d3eec66ba92ee31e87edf0daebb6c xor 817c7fad27fbd3af70a93f9e4ceffa2d = 41414141414141414141414141414141
8d6d224eb18cfa6f069e3d05809c1c33 xor cc2c630ff0cdbb2e47df7c44c1dd5d72 = 41414141414141414141414141414141
9eed2e770495fd97bbc32331ff5be9ac xor dfac6f3645d4bcd6fa826270be1aa8ed = 41414141414141414141414141414141
766d9a71b27e0cae25985f7515b61f34 xor 372cdb30f33f4def64d91e3454f75e75 = 41414141414141414141414141414141
cc41368977dd306a590be501aabba758 xor 8d0077c8369c712b182272af1eda3a63 = 41414141414141412997aeb4619d3b
22cdbba16f4d65aa8ce852728118766b xor 92b0c90883cfec10e5b85439a30c4a06 = b07d72a9ec8289ba6950064b22143c6d
cf5169fe6fdd622a352da56075f96443 xor aa221a9f08b85c7d5a5a8940128b0122 = 6573736167653e576f772c2067726561
e8c554201148d4285cd932fb12da7ae2 xor 9ce5314e723aad5828b05d9533e6558f = 7420656e6372797074696f6e213c2f6d
83903db3d4425c7f72db03c4edecf044 xor e6e34ed2b3276275f8b5a8011b3e00db = 6573736167653e0a8a6eabc5f6d2f09f
d46ab920f2061780853b74d3a012c54c xor c957c6ddc24b827e4d4af69a5df2b71e = 1d3d7ffd304d95fec8718249fde07252
3e0ad842f266d0d08f2619ab9534995a xor 500540da22a095d0e6956a38e08a6c4d = 6e0f9898d0c6450069b3739375bef517
f391677976df6366e89a0ae96a49     xor ea181681ec5d73332f164fe154690d29 = 198971f89a821055c78c45083e20
```

MD5 Stream Cipher

1. Plaintext erraten (0-255)
2. Keystream in Ciphertext auffinden
3. Restliche Nachricht entschlüsseln
4. <message> auffinden