Twenty-Sixth Annual

Willamette University—TechStart High School Programming Contest

Saturday 3 March 2012

OFFICIAL CONTEST TEST DATA

COURTESY COPY FOR TEAMS—OK TO DISTRIBUTE!!

General advice to judges

- always read the problem before testing a program (if you haven't read the problem previously)
- allow for slight variations in how output is presented, but not for substantive issues in the answer (if you're not sure which is which, ask the Contest Director, i.e., Fritz)
- **allow for reasonable user interfaces**, as long as they are clearly labeled, or prompt appropriately for inputs
- remember to include any promised sentinel values (e.g., 0, -1, XX, a blank line) to terminate input
- if a solution fails, make some effort to see why (hit extra carriage returns, etc.—read the code!)
- in general, we don't give hints about why a solution failed, but in persistent "hard luck" cases, we will
- if a program runs correctly for all test inputs, but you suspect it is still wrong, we can add new test data
- remember, the contest is supposed to be fun (and educational)!

1. **Change is in the air** *(input is 5 integers; output is change-making sentence or 'not possible')*

   (Don't fuss over singular/plural)

```
Input:   87  4  3  3  2
Output:  Make 87 cents with 2 quarters, 3 dimes, 1 nickel and 2 pennies.

Input:   87  4  2  5  4
Output:  Make 87 cents with 2 quarters, 2 dimes, 3 nickel and 2 pennies.

Input:   118  4  2  2  4
Output:  Make 118 cents with 2 quarters, 1 dime(s), 1 nickel and 3 pennies.

Input:   29  3  3  0  4
Output:  Not possible!
```

2. **What's the frequency, Kenneth?** *(input is a string, mixed case, possible punctuation; no 0 frequencies, please!)*

```
Input?     The rains in Spain run mainly down the drain.
Output:    7n 5i 4a 3r 2d 2e 2h 2s 2t 1l 1m 1o 1p 1u 1w 1y

Input?     The blintz in Minz spins tinsely in the rinse.
Output:    7i 7n 4e 4s 4t 2h 2l 2z 1b 1m 1p 1r 1y
```

```
Input?      X, Y , ..., Z
Output:     1x 1y 1z

Input?      aaaaaaaaaaaaaaa bbbbbbbbbb AAAAA      (15 as, 10 bs, 5 As)
Output:     20a 10b
```

3. Multimedia maven *(one float, then lines of names & floats)*

Input:

```
6.0
hoo    420.0
goo    240.0
foo    540.0
zoo    360.0
yoo    180.0
too    480.0
```

Output:

```
yoo    @    03:00
goo    @    03:50
zoo    @    05:10
hoo    @    05:40
too    @    06:00
foo    @    06:10
```

Input:

```
4.0
peter  740.0
micky  540.0
mike   840.0
davy   140.0
```

Output:

```
davy   @    02:20
micky  @    07:20
peter  @    09:00
mike   @    09:25
```

4. Taxation vexation *(one non-negative integer, then two floats (2 digits post-point); result should be float)*

```
Number of items? 10
Total price? 10.85
Average pre-tax item cost was: 1.00

Number of items? 5
Total price? 10.91
Average pre-tax item cost was: 2.01

Number of items? 3
Total price? 8.14
Average pre-tax item cost was: 2.5

Number of items? 1
Total price? 1.08
Average pre-tax item cost was: 1.00
```

5. Sorting by hand *(input and output are both sequences of card codes; 'x' means ten!)*

```
Input:    9s   jd   9c   7h   7c   jc
Output:   jc   9c   7c   jd   7h   9s

Input:    3h   4d   5s   kh   qd   7c   9d   4h   as
Output:   7c   4d   9d   qd   4h   kh   3h   as   5s

Input:    as   5s   ks   ns   qs   3s   ac
Output:   ac   as   5s   ks   ns   qs   3s
```

6. Binary palindromes *(non-negative integers, one per line; sentinel is -1; **must show binary in their output**)*

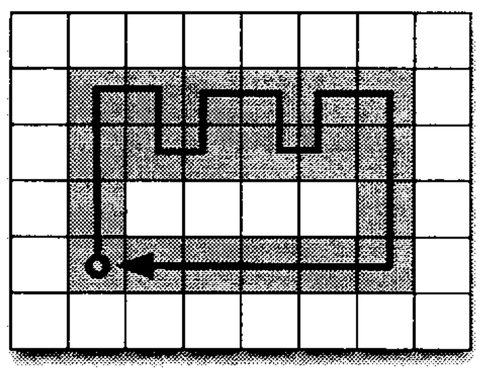
```
102          binary: 1100110                  No
33           binary: 100001                   Yes
123456789    binary: 111010110111100110100010101      No
1193         binary: 10010101001              Yes
0            binary: 0                        Yes      (blank output not acceptable)
1            binary: 1                        Yes
3326163      binary: 1100101100000011010011   Yes
6439         binary: 1100100100111            No       (extra 1 on right)
2015         binary: 11111011111              Yes
975          binary: 1111001111               Yes
3919         binary: 111101001111             No       (2 zeroes right of middle)
```

7. Martian meanderings *(input is N-E-S-W path; output labelled somehow, 3 integers)*

```
Input:    N N W N   S E E N   E E W W   S S S W
Output:  length: 16; area: 11; edges: 20

Input:    N N N E   E S S E   E E N N   W W W S   S S W W
Output:  length: 20; area: 17; edges: 32

Input:    N N N  E S W N  E S W N  E  S S S  W W W W W
Output:  length: 20; area: 20; edges: 30
```



8. Web spinner *(pairs of floats, space-separated; output includes letters & sometimes numbers!)*

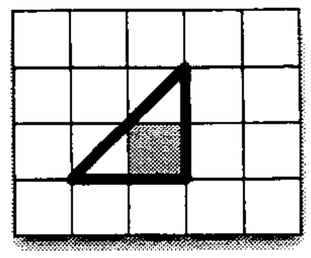
Note! *many different solutions are possible; check path length first*

```
Input:   4.0  8.6   5.0  2.5   2.0  5.5
Output: 5 O C  C C O O C  C C C O C  I I I I I

Input:   1.0  9.0   1.0  0.0   1.0  5.0  1.0  4.0
Output: 9 C I 5 X I

Input:   6.0  7.0   6.0  6.0   6.0  3.0  6.0  2.0
Output: 7 O O O O O X X X X X I I I I I I
```

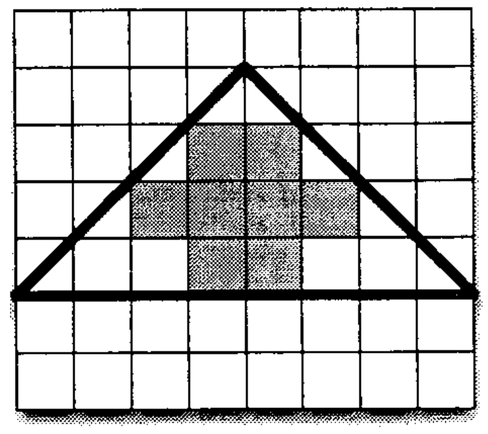
9. Triangular hull *(series of pairs of integer numbers, space-separated; Perimeter in output!)*



Note! *any rotated or flipped version is OK; concentrate first on the perimeter (and check it!)*

Input: 2 1

Output: 1 1 3 1 3 3 Perimeter = 6.8



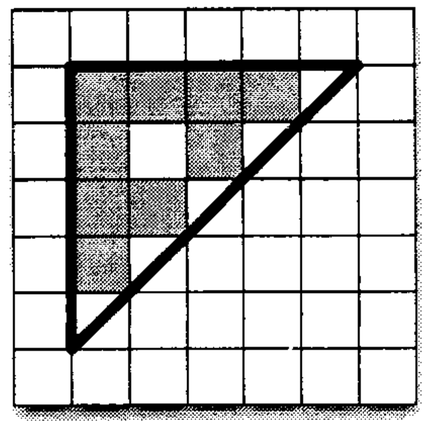
(from problem sheet)

Input: 3 3 3 4 3 5 2 5 2 6

Output: 4 1 4 7 1 7 Perimeter = 15.7

Input: 2 3 3 3 4 3 5 3 3 2 4 2 3 4 4 4

Output: 0 2 8 2 4 6 Perimeter = 19.31



Input: 1 2 1 3 1 4 1 5 2 3 2 5 3 4 3 5 4 5

Output: 1 1 1 6 6 6 Perimeter = 17.07

10. Words within words *(lines of text with 2 words each; no clear termination)*

spy	spray	Yes / … IS in …
bond	blonde	Yes / … IS in …
bond	trouble	NO / … is NOT in …
bonde	bond	NO / … is NOT in …
tile	reticulated	Yes / … IS in …
a	a	Yes / … IS in …
aa	a	NO / … is NOT in …
a	dementia	Yes / … IS in …
art	arthropod	Yes / … IS in …
ball	babbling	NO / … is NOT in …

11. Operation insertion *(integers on a line, at least 2, no more than 10;* ***other answers may be possible!)***

Should work for just 2 numbers:

Input: 5 5

Output: 5 = 5

Even when no equation is possible:

Input: 5 10

Output: No equation is possible.

Problem sheet sample data:

```
Input:  1  2  3  10  3

Output: 1 + 2 * 3 = 10 - 3
```

Problem sheet sample data:

```
Input:  1  5  8

Output: No equation is possible.
```

Test association of plus/minus:

```
Input:  10  2  3  11

Output: 10 - 2 + 3  =  11
```

Multiple answers possible: they need only return one:

```
Input:  5  5  5  5  5  5

Output:
        5 + 5 + 5 = 5 + 5 + 5

        5 * 5 + 5 = 5 * 5 + 5

    etc.
```

12. Traffic turmoil *(car labels lowercase, directions uppercase, times are mins & secs.)*

Input:

```
a  N  5:00
c  S  5:20
d  W  5:20
b  E  5:20
p  E  5:25
q  E  5:35
u  N  7:00
```

Output:

```
a  5:10
b  5:30    (b gets priority here, based on direction)
c  5:40
d  5:50
p  6:00    (p had to wait for his turn "round robin")
q  6:10    (q had to wait for his turn behind p)
u  7:10    (u must arrive before she leaves!)
```

Input:

```
j  W  7:00
k  S  7:05
m  S  7:06
n  S  7:07
o  S  7:08
p  S  7:09
z  N  7:55
```

Output:

```
j  7:10
k  7:20
m  7:30
n  7:40
o  7:50
p  8:00
z  8:10
```

Input:

```
a   W   3:00
b   S   3:00
c   E   3:00
d   N   3:00

h   N   4:00
i   E   4:01
j   S   4:02
k   W   4:03

w   W   4:05
x   S   4:06
y   E   4:07
z   N   4:08
```

Output:

```
d   3:10
c   3:20
b   3:30
a   3:40

h   4:10
i   4:20
j   4:30
k   4:40

z   4:50
y   5:00
x   5:10
w   5:20
```

13. Freaks, geeks and cliques *(input one per line, letters lowercase; colon separator, **sometimes blank after that!**)*

Input:

```
a: b c
t: f
d: p
m: n
p:
q: p
n: a
b: a z
z:
f: t
c: a z
```

Output: *(in any order, cliques* and *members)*

```
Clique #1:  q d p
Clique #2:  m c a n b z
Clique #3:  f t
```

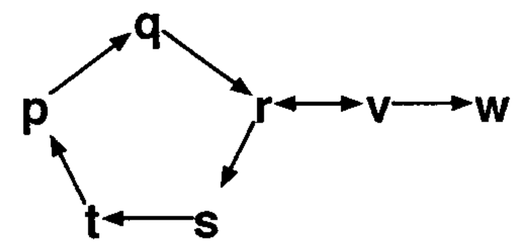
Input:

```
a:
b:
p: q
q: r
r: s v
s: t
t: p
v:r w
w:
```

Output: *(in any order, cliques* and *members)*

```
Clique #1:  a
Clique #1:  b
Clique #3:  p q r s t v w
```

a b

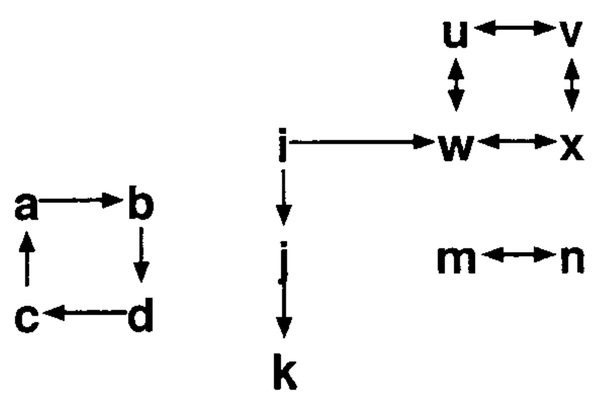


Input:

```
a: b
b: d
d: c
c: a
i: j w
j: k
k:
m: n
n: m
u: v w
v: u x
w: x u
x: w v
```

Output: *(in any order, cliques* and *members)*

```
Clique #1:  a b c d
Clique #2:  m n
Clique #1:  i j k u v w x
```



14. Doing it with class! *(input class defs first, then variable/accessor expressions; UPPER/lowercase matters!)*

Input:

```
class A { i x; i y; C p; }
class B > A  { d z; A q; }
class C { d x; B r; C s; }
```

```
var B b: b.q.y                    Output:  Valid with type i
var C h: h.r.p.s.x                Output:  Valid with type d
var A m: m.p.r.q.z                Output:  Invalid!
var B u: u.q                      Output:  Valid with type A
```

Input:

```
class P { d a; Q b; P c; }
class Q > P  { b j; P k; }
class M > Q  { M u; i v; P w; }
```

```
var Q q: q.j                      Output:  Valid with type b
var Q q: q.b.c                    Output:  Valid with type P
var P x: x.c.c.c                  Output:  Valid with type P
var M n: n.u.k.b.b                Output:  Valid with type Q
var M z: z.b.w.a                  Output:  Invalid!
```