# Open Programming Competition Sample
# Department of Math and CS
# UMSL

## Problem 1 : Wrapping Numbers Detection

A cyclic number is an integer n digits in length which, when multiplied by any integer from 1 to n, yields a "wrapping" of the digits of the original number. That is, if you consider the number after the last digit to "wrap around" back to the first digit, the sequence of digits in both numbers will be the same, though they may start at different positions. For example, the number 142857 wraps around, as illustrated by the following table:

```
142857      * 1     =       142857
142857      * 2     =       285714
142857      * 3     =       428571
142857      * 4     =       571428
142857      * 5     =       714285
142857      * 6     =       857142
```

## Input

Write a program which will determine whether or not a number is wrapping. The program should take as a command line argument an integer from 2 to. (Note that preceding zeros should not be removed, they are considered part of the number and count in determining n. Thus, "01" is a two-digit number, distinct from "1" which is a one-digit number.)

```
prompt$ prog1 142857
```

## Output

Your program should output either wrap or not wrap.

## Sample Test Cases

| Invocation | Output |
|---|---|
| 142857 | wrap |
| 142856 | not wrap |
| 142858 | not wrap |
| 01 | not wrap |
| 0588235294117647 | wrap |

## Problem 2 : Union and Intersection

The union of two sets is the set containing all the elements of both of those sets. The intersection of a set is the set containing elements that were in both sets. In this problem you will be writing a program to calculate the union and intersection of two sets. Recall that a set does not allow duplicates and that the order of the elements does not matter in a set.

## Input

Your program will take as command line arguments two sets of integers. The format of this data is the number of elements in set one, followed by those number of elements. Then the size of the second set, followed by all of its elements. If one of these sets is empty, it would be represented by a size of 0 and no elements after it. The elements in these sets are not necessarily sorted

```
prompt$ prog2 3 3 4 7 2 9 3
```

## Output

The output will be the size of the set formed by the union of both sets, followed by the union of both sets, followed by the size of the set formed by the intersection of both sets, followed by all the elements of the intersection of both sets. *The output of these sets should be in order.* Remember when testing that a set could have 0 elements, so the input could be some combination of empty sets. The empty set would be represented by 0, followed by no other input.

## Sample Test Cases

| Invocation | Output |
|---|---|
| 3 3 4 7 2 9 3 | 4 3 4 7 9 1 3 |
| 0 0 | 0 0 |
| 1 3 0 | 1 3 0 |
| 0 1 3 | 1 3 0 |
| 3 1 5 2 3 2 5 1 | 3 1 2 5 3 1 2 5 |

## Problem 3 : Number Combining

You have two files containing integer numbers separated by white spaces (spaces, tabs, new lines). You also have an integer number you are looking for.
The objective is to find out if any, and how many, combinations of numbers (one from each file, order irrelevant) when put together end up to be the third number.
Negations are processed separately following the rule of multiplication (2 negations make the resulting number positive).
For example if the first file contains: -27 23 -1 356 2
and the second file contains: 56 27 1 2 -71
and the number to look for is: -271
then the number can be obtained by putting together 2 from the first file with -71 from the second file, or from putting together -27 from the second file with 1 from the first file.
So there are two ways to do that (two cases).
Note the sought number is negative so one of the two numbers put together must be negative.

### Input

Two filenames containing numbers, followed by a number. Note that the filenames can be arbitrary, and the number can be a positive or negative integer. You may assume each file contains unique integers only, and the third argument is also an integer. Example:

```
prompt$ prog3 fileOne.txt fileTwo.txt -372
```

### Output

If any of the files cannot be open you print

```
ERROR
```

Otherwise print the message as below followed by the number of cases found (a case is a pair of numbers, one from each file, that when put together will give the sought number). Below assume 2 pairs matched the number

```
MATCH 2
```

### Notes

- The found pair must come from both files (cannot be just a number from one file) and the order is not relevant.
- No spaces in output, except for one space between MATCH and number.
- When seeking a negative number, one of the selected numbers must be negative. When seeking a positive number, the selected numbers must be both positive or both negative.

### Sample test cases

Assume fileX and fileY are two files containing signed integers.
fileX: 1  2  3 -1 -2 -3 26 15
fileY: 5  6  7 -5 -6 -7 67 -67 15

| Invocation Parameters | Output |
|---|---|
| fileX file 234 | ERROR |
| fileX fileY 151 | MATCH 0 |
| fileX fileY 267 | MATCH 2 |
| fileY fileX 267 | MATCH 2 |
| fileX fileY 15 | MATCH 2 |
| fileX fileY -152 | MATCH 1 |

### Problem 4 : Applied Queueing

Patients going to the Rocky Urgent Care enter a queue to be seen by a doctor, i.e. the person who enters first is seen first. However, sometimes, when a new patients arrives, the receptionist may decide to put this patient to the front of the queue, instead of the end. From a computer simulation perspective, there are three operations that can be performed in this patient queue:
1) enqueue a new patient at the end of the queue
2) enqueue a patient at the front of the queue
3) dequeue or see a patient and remove him/her from the queue.

### Input

A text file with a series of these three operations. For example, suppose we have a text file called patientData.txt containing:
enqueue front Bob
enqueue end Marie
enqueue front Charlie
enqueue front Charles
dequeue
dequeue
Then the program would be run as

```
prompt$ prog4 patientData.txt
```

### Output

Print the resulting queue in comma separated format. For example, the queue output for the previous file will be “Bob,Marie”.

### Note

You may assume the input file is well-formed, though it may be read-only.

## Problem 5 : ASCII Pathing

Write a program that will read data from a file. The filename is entered through the command line. The data contains character strings of 1 to 80 characters. The strings will vary in length. There will be at least one such string, and there may be as many as 100. Each character will be an upper case letter from the set {N,S,E,W}.
Assume that a marker is set on the origin (0,0) of a two dimensional Cartesian graph. Each time a character is read from the file, the marker should move. If the character read is N, the marker should move up one unit; if S, down one unit; if E, right one unit; and if W, left one unit.
When all the characters are read from the file, your program should output 5 coordinates, one after another on the same line, each in the format (X,Y), where X and Y are integers. The first coordinate is the final position of the marker. The second coordinate should be the highest position the marker ever occupied. The third coordinate should be the rightmost position the marker ever occupied. The fourth coordinate should be the lowest position the marker ever occupied. The fifth coordinate should be the leftmost position the marker ever occupied. If there are multiple points that qualify for one of these points, then chose the one that was visited first by the marker.

### Input

Filename should be supplied as a command line argument. Inside the data file, there are one or more lines of characters, each line from 1 to 80 characters inclusive. Each character in the strings is from the set {N,S,E,W}. End-of-line symbols are to be ignored. If there are any characters other than specified, the program should end by printing an error message "ERROR".

```
prompt$ prog5 mydata.txt
```

File `mydata.txt` contains character strings. Example

```
NSEWN
NNEEWNNSN
```

### Output

For the data file above, using the rules the output should be:
(1,5)(1,5)(2,3)(0,0)(0,0)

### Notes

- If no command line argument is provided, print `ERROR` on the screen.
- If the file cannot be opened or if it is empty, print `ERROR` on the screen.

### Sample test cases

| Inputs | Output |
|---|---|

| | |
|---|---|
| File not readable | ERROR |
| S | (0,-1)(0,0)(0,0)(0,-1)(0,0) |
| NSEWSNWE | (0,0)(0,1)(1,0)(0,-1)(-1,0) |
| File empty | ERROR |
| No filename | ERROR |

## Problem 6 : Intersecting the Axis

In this problem you will be working with the coordinate plane. On a coordinate plane, there are two points A and B with integer coordinates that are above the x-axis. Your program should find a point C on the x-axis with integer coordinates as well, such that AC + BC is the minimum.

### Input

Input is four integers, the x,y coordinates of point A and the x,y coordinates of point B.

```
prompt$ prog6 3 7 10 18
```

### Output

The resulting Y coordinate of the point C.

### Sample test cases

| Inputs | Output |
|---|---|
| 1 1 3 2 | 2 |
| 2 1 5 5 | 3 |

## Problem 7 : Thinking outside of the Box

In this program you will read a file that is holding an $m \times n$ integer matrix. Your program should search this $m \times n$ matrix for a $2 \times 2$ matrix that sums to 0. Once you have found that matrix, return the sum of integers just outside of that square. Note that we have periodic boundary conditions on this problem, so if the square matrix is at the side of the integer matrix, you would sum up elements on the other side.

## Input

The filename is supplied as a command line argument. Inside the data file, the integer entries are separated by a space. Each line corresponds to one row in the matrix. You can assume that a valid integer matrix is stored in the data file.

```
prompt$ prog7 intmatrix.txt
```

## Output

The sum of the elements just outside the found 2x2 matrix that summed to 0.

For example, if called with the matrix

| | | | | |
|---|---|---|---|---|
| 5 | -3 | 27 | 8 | 11 |
| 15 | 0 | -2 | 6 | -2 |
| 1 | 3 | 18 | 4 | -8 |
| 11 | 18 | 36 | 17 | 2 |

Then we would find the largest square as the ones highlighted in yellow and the entries that should be summed (27 + 8 + 11 + 5 + -2 + 18 + 36 + 17 + 2 + 11 + 15 + 1) are in red. So the correct output would be 149.

## Problem 8 : Crypto-arithmetic

You are given three five-character strings separated by a space. The strings contain largely numbers but also three place-holder variables: $A$, $B$, and $C$. Your job is to find the numeric value corresponding to those three variables such that the sum of integers, after substitution of variables, in the first string and the second string equals the third string. Each string contains at least one of the three variables, possibly more.

## Input

Three command line arguments that are of five characters each. Example:
C793A B83B5 4ABAC
This would be given to your program as follows:

```
prompt$ prog8 C793A B83B5 4ABAC
```

## Output

Print the values of the three variables, separated by a space, in the order of A, B and C.
Given the previous input, 17936 + 28325 = 46261, so the output should be:
6 2 1