bc - an arbitrary precision calculator language

```
Flags:
        -l : uses mathlib libraries and makes more functions available
        -q : quiet, doesn't show headers when starting

Commands:
        scale : changes how many decimal places to use (for integer math, set to 0)
        ibase : this is the base numbering system for input
        obase : this is the base numbering system for output
        last : this returns the last outputted number

Example:
        basic usage

$ bc -q -l
/* -l command loads mathlib and sets scale=20 */
scale
20
3+4
7
4*5
20
/* a(x) is the arctan(x), we can use it to define pi */
pi=4*a(1)
radius=7
circumference=2*pi*radius
area=pi*radius^2
pi
3.14159265358979323844
radius
7
circumference
43.98229715025710533816
area
153.93804002589986868356
/* if we have three circles, how much total area is it */
last*3
461.81412007769960605068
/* if you use modulus, watch your scale */
scale
20
10%6
.00000000000000000004
scale=0
10%6
4

Example:
        convert a hexadecimal number into decimal and binary.
        Note, hexadecimal characters in bc have to be capitalized

$ bc -q
ibase=16
6F1F767BF5E14A4DE9D5DF
1343393449862866403313474 23
obase=2
6F1F767BF5E14A4DE9D5DF
110111100011111011101100111101111110101111000010100101001001101111 01\
0011101010111011111
```

Example:
Run our RSA algorithm and encrypt/decrypt the number 17. Our modulus is 55, our public exponent is 7 and our private exponent is 23. The modulo operator is "%", the "^" operator is used for exponentiation.

```
$ bc -q
17^7%55
8
8^23%55
17
```

openssl – certificate swiss army knife

```
openssl command [ command_opts ] [ command_args ]

Commands:
        x509           : give us information about a certificate file
        rsa            : give us information about a key file
        genrsa         : generate an RSA key
        s_client       : connect to a host port and talk TLS/SSL.  Supports plain SSL,
                         or TLS for smtp, pop3, imap, ftp and ldap (requires patch)
        dgst           : run a cryptographic digest like SHA256 or MD5
        aes-256-cbc    : encrypt using AES
```

Example:
open an x509 certificate and list it's contents. If a file contains multiple certificates, only the first is shown

```
$ openssl x509 -in /etc/pki/tls/certs/server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            60:01:97:b7:46:a7:ea:b4:b4:9a:d6:4b:2f:f7:90:fb
         .........
```

Example:
generate and display a 31bit RSA key. This is an example of using both the "genrsa" and "rsa" commands together with a pipe on the command line.

```
$ openssl genrsa 174 | openssl rsa -text -noout
Generating RSA private key, 174 bit long modulus
..++++++++++++++++++++++++++
...++++++++++++++++++++++++++
e is 65537 (0x10001)
Private-Key: (174 bit)
modulus:
    2a:d3:21:08:47:24:fe:33:fd:d1:06:f7:f1:eb:fd:
    23:5b:cd:05:c6:77:33
publicExponent: 65537 (0x10001)
privateExponent:
    1f:ed:b5:ad:14:4e:12:70:f5:06:48:cd:5f:88:2f:
    ef:08:bc:ce:93:92:b1
prime1:
    6f:1f:76:7b:f5:e1:4a:4d:e9:d5:df
prime2:
    62:a8:5e:84:80:df:93:94:ea:01:2d
exponent1:
    24:22:d7:24:f8:90:8a:d9:4b:71:81
exponent2:
    01:9d:e4:be:a3:26:06:d3:1f:ea:99
coefficient:
    5f:51:65:13:e5:f2:64:5b:7f:c3:a2
```

Example:
multiple the above prime numbers together to verify the modulus. Notice the order of the “obase” and “ibase”. If you switch them, you are setting your output base to 0x16, or 22 in decimal. This also should be on one line.

```
$ echo "6f:1f:76:7b:f5:e1:4a:4d:e9:d5:df * 62:a8:5e:84:80:df:93:94:ea:01:2d" | \
  sed 's/://g' | \
  tr 'a-f' 'A-F' | \
  sed 's/^/obase=16;ibase=16;/'| \
  bc | \
  tr 'A-F' 'a-f' | \
  sed 's/\(..\)/\1:/g'
2a:d3:21:08:47:24:fe:33:fd:d1:06:f7:f1:eb:fd:23:5b:cd:05:c6:77:33:
```

Example:
connect to a remote host and look at the certificates it has to offer. The “showcerts” option will display all the certs offered by the server. These can be examined with the x509 command. Also notice, once connected, we can interact with this host as if we just used telnet to port 80. This is extremely useful for troubleshooting what certificates a host offers, and how to interact over an encrypted SSL tunnel.

```
$ openssl s_client -connect security.ias.edu:443 -showcerts
...
Certificate chain
 0 s:/serialNumber=s21oo4hS1uaOFxYSUPk-iXFVLhqbRJor/OU=GT09845384/OU=See www.rapidssl.com/resources/cps
(c)12/OU=Domain Control Validated - RapidSSL(R)/CN=*.ias.edu
   i:/C=US/O=GeoTrust, Inc./CN=RapidSSL CA
-----BEGIN CERTIFICATE-----
MIIFITCCBAmgAwIBAgIDCLRaMA0GCSqGSIb3DQEBBQUAMDwxCzAJBgNVBAYTAlVT
MRcwFQYDVQQKEw5HZW9UcnVzdCwgSW5jLjEUMBIGA1UEAxMLUmFwaWRTU0wgQ0Ew
...
Lrqc5UefRIeNq9fehJHIyqBoV/UB9E91OgvJR5DUXrWh89XzSWCcvih9nKGWb+nT
xeliFSVdjczHc+qZF9D4CZ+mt9YC
-----END CERTIFICATE-----
 1 s:/C=US/O=GeoTrust, Inc./CN=RapidSSL CA
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIDAjbRMA0GCSqGSIb3DQEBBQUAMEIxCzAJBgNVBAYTAlVT
MRYwFAYDVQQKEw1HZW9UcnVzdCBJbmMuMRswGQYDVQQDExJHZW9UcnVzdCBHbG9i
YWwgQ0EwHhcNMTAwMjE5MjI0NTA1WhcNMjAwMjE4MjI0NTA1WjA8MQswCQYDVQQG
...
knYYCnwPLKbK3opie9jzzl9ovY8+wXS7FXI6FoOpC+ZNmZzYV+yoAVHHb1c0XqtK
LEL2TxyJeN4mTvVvk0wVaydWTQBUbHq3tw==
-----END CERTIFICATE-----
...
---
```

GET / HTTP/1.0

```
HTTP/1.1 200 OK
Date: Tue, 19 Nov 2013 18:50:20 GMT
Server: Apache/2.2.15 (Red Hat)
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.4.26.4-3396
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
Content-Length: 185
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
...
```

Example:
Calculate the SHA256 sum on a host that only has openssl and not the sha256sum tool.

```
$ sha256sum /etc/hosts
bash: sha256sum: command not found...
$ openssl dgst -sha256 /etc/hosts
SHA256(/etc/hosts)= 42c60aee9ac2254ea7216735923816491448066 9c06c2fad31123344fe71a7f
```

Example:
Quickly encrypt a config file to send over email to a vendor for troubleshooting purposes. I do this all the time, it isn't too difficult to explain over the phone how to decrypt, and it gives you the option of protecting sensitive data over an insecure medium. You still have to tell them the password over the phone, though, which is better than sending cleartext.

```
$ openssl aes-256-cbc -in database_credentials.php -out database_credentials.php.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:

$ file database_credentials.php*
database_credentials.php:     ASCII text
database_credentials.php.enc: data

$ more database_credentials.php
<?php

$dbuser = "dummy";
$dbpass = "drowssap";
$dbhost = "hopeyoudonthackme.com";
$dbname = "please";

?>

$ xxd database_credentials.php.enc
0000000: 5361 6c74 6564 5f5f 70d2 c0dc c413 bfbf  Salted__p.......
0000010: a7e0 124e 6477 42e7 b553 17ff ee6c edb4  ...NdwB..S...l..
0000020: ab5b 15b2 ab0a 455c d2ef 0cb2 e87a 8350  .[....E\.....z.P
0000030: 2fe4 9b6a 8910 5e8a 2b56 56ce 8f5c c727  /..j..^.+VV..\.'
0000040: ebae 66b2 1218 f4fc 2c18 e375 f45d 4915  ..f.....,..u.]I.
0000050: a756 1d1d bdb2 a4ab 0b1f 844b 4fe9 7752  .V.........KO.wR
0000060: bdba c4a7 27f1 f965 0b19 3370 74cc beb3  ....'..e..3pt...
0000070: 25ad 5c94 bbb9 8581 b36e ffd1 6301 2b59  %.\......n..c.+Y

$ base64 < database_credentials.php.enc | \
mail -s "Database credentials you asked for" support@example.com
```

Example:
Inspect an ssh RSA key for it's components

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ep/.ssh/id_rsa): /home/ep/.ssh/njedge
...
Your identification has been saved in /home/ep/.ssh/njedge.
Your public key has been saved in /home/ep/.ssh/njedge.pub.
...

$ openssl rsa -in .ssh/njedge -text -noout
Private-Key: (2048 bit)
modulus:
    00:e7:4e:c9:dc:0e:6a:22:f0:ca:48:c7:ea:6b:a9:
...
```