Can load this from http://www.math.sunysb.edu/~scott/mat331.spr13/daily/2013-03-04.mw
(or just go to class web page)

```
> with(StringTools):
> Alphabet:=cat("\n",Select(IsPrintable,convert([seq(i, i=1..255)],
  bytes)));
```

Alphabet := "
 !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
    ^_`abcdefghijklmnopqrstuvwxyz{|}~"   **(1)**

```
> StringToList := proc(text::string)
     local i;
    global Alphabet;
    [seq(SearchText(text[i],Alphabet)-1,i=1..length(text))];
  end:
  ListToString := proc(numlist::list(nonnegint))
     local i;
    global Alphabet;
    cat(seq(Alphabet[numlist[i]+1],i=1..nops(numlist)));
  end:
> Caesar:=proc( text::string, shift::integer)
    local numlist, codenum,p;
    global Alphabet;
    p:=length(Alphabet);
    numlist:=StringToList(text);
    codenum:=[seq( modp(numlist[i]+shift,p),i=1..length(text))];
    return(ListToString(codenum));
  end:
> crypt:=Caesar("Et tu, Brute?",12);
```

crypt := "Q , !8,N~! qK"   **(2)**

```
> Caesar(crypt,-12);
```

"Et tu, Brute?"   **(3)**

instead of x -> x+shift, use x -> a*x+shift

```
> Affine:=proc( text::string, a::integer, shift::integer)
    local numlist, codenum,p;
    global Alphabet;
    p:=length(Alphabet);
    numlist:=StringToList(text);
    codenum:=[seq( modp(a*numlist[i]+shift,p),i=1..length(text))];
    return(ListToString(codenum));
  end:
> crypt:=Affine("Jimmy-Jimmy Bobo", 1, 5);
```

crypt := "Onrr~2Onrr~%Gtgt"   **(4)**

```
> Affine(crypt,1,-5);
```

"Jimmy-Jimmy Bobo"   **(5)**

```
> crypt:=Affine("Jimmy-Jimmy Bobo", 11, 5);
```

crypt := "}R~~B^}R~~B/%4e4"   **(6)**

```
> Affine(crypt, 1/11, -5);
Error, invalid input: Affine expects its 2nd argument, a, to be
of type integer, but received 1/11
> y=11*3 + 5 mod 96;
```

$y = 38$ **(7)**

```
> (38-5)/11 mod 96;
                                        3                                    (8)
> 1/11 mod 96;
                                       35                                    (9)
> 35*11;
                                      385                                   (10)
> 385 mod 96;
                                        1                                   (11)
> evalf(1/11);
                                 0.09090909091                              (12)
> 35*% mod 96;
Error, invalid argument for modp or mods
> AffInv:=proc( text::string, a::integer, shift::integer)
    local numlist, codenum,p, ainv;
    global Alphabet;
    p:=length(Alphabet);
    ainv:=modp(1/a,p);
    numlist:=StringToList(text);
    codenum:=[seq( modp(ainv*(numlist[i]-shift),p),i=1..length
  (text))];
    return(ListToString(codenum));
  end:
> AffInv(crypt,11,5);
                              "Jimmy-Jimmy Bobo"                            (13)
> crypto:=Affine("How do I work this?", 6, 17);
                        crypto := "f0`6N06l6`0Bx6NflH0"                     (14)
> AffInv(crypto,6,17)
Warning, inserted missing semicolon at end of statement
Error, (in AffInv) the modular inverse does not exist
> 1/2 mod 96;
Error, the modular inverse does not exist
> msolve( 2*x = 1, 96);
> msolve( 11*x = 1, 96);
                                   {x = 35}                                 (15)
> msolve(25*x=1,96);
                                   {x = 73}                                 (16)
> msolve(15*x=1,96);
> ifactor(96);
                                  (2)^5 (3)                                 (17)
> isprime(97);
                                     true                                   (18)
> Alphabet:=cat("\n\t",Select(IsPrintable,convert([seq(i, i=1..255)
  ], bytes)));
                                Alphabet := "                               (19)
             !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
     [\]^_`abcdefghijklmnopqrstuvwxyz{|}~"
> length(Alphabet);
                                      97                                    (20)
```

```
> crypto:=Affine("How do I work this?", 6, 17);
                        crypto := "i0`;O00;o;`0By;NgmH3"                        (21)
> AffInv(crypto,6,17);
                              "How do I work this?"                              (22)
> crypto:=Affine("How do I work this?", 97, 19);
                       crypto := "1111111111111111111"                         (23)
> breakme:=Affine("How do I work this?", 47, 19);
                       breakme := "SI=.)I.!.=IuO.r#RC0"                        (24)
> StringToList(" ?");
                                    [2, 33]                                    (25)
> StringToList(".0");
                                   [16, 18]                                    (26)
> msolve( {a*2+b=16, a*33+b=18}, 97);
                                {a = 47, b = 19}                               (27)
> msolve( {A*16+B=2, A*18+B=33}, 97);
                                {A = 64, B = 45}                               (28)
> Affine(breakme,64,45);
                              "How do I work this?"                              (29)
> Affine("abcdefghijk", 5, 7);
                                "QV[`ejoty~""                                   (30)
> L:=StringToList("I'm not a vector");
          L := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86, 81, 84]     (31)
> seq([L[i],L[i+1]], i=1..nops(L)-1);
[43, 9], [9, 79], [79, 2], [2, 80], [80, 81], [81, 86], [86, 2], [2, 67], [67, 2], [2, 88], [88,   (32)
    71], [71, 69], [69, 86], [86, 81], [81, 84]
> [seq([L[i],L[i+1]], i=1..nops(L)-1,2)];
       [[43, 9], [79, 2], [80, 81], [86, 2], [67, 2], [88, 71], [69, 86], [81, 84]]      (33)
> StringTo2Vect:=proc( s::string)
    local L;
    L:=StringToList(s);
    [seq([L[i],L[i+1]], i=1..nops(L)-1,2)];
  end:
> StringTo2Vect("I'm not a vector");
       [[43, 9], [79, 2], [80, 81], [86, 2], [67, 2], [88, 71], [69, 86], [81, 84]]      (34)
> StringTo2Vect("I'm not a vector!");
       [[43, 9], [79, 2], [80, 81], [86, 2], [67, 2], [88, 71], [69, 86], [81, 84]]      (35)
> StringTo2Vect:=proc( s::string)
    local L;
    if ( modp(length(s),2) <> 0) then
         s:=cat(s,"X");
     fi;
    L:=StringToList(s);
    [seq([L[i],L[i+1]], i=1..nops(L)-1,2)];
  end:
> StringTo2Vect("I'm not a vector!");
Error, (in StringTo2Vect) invalid left hand side in assignment
> s:="mama";
                                  s := "mama"                                  (36)
```

```
> s:=cat("baby",s);
                              s := "babymama"                                  (37)
> StringTo2Vect:=proc( t::string)
    local L, s;
    s:=t;
    if ( modp(length(s),2) <> 0) then
         s:=cat(s,"X");
     fi;
    L:=StringToList(s);
    [seq([L[i],L[i+1]], i=1..nops(L)-1,2)];
  end:
> V:=StringTo2Vect("I'm not a vector!");
  V := [[43, 9], [79, 2], [80, 81], [86, 2], [67, 2], [88, 71], [69, 86], [81, 84], [3, 58]]   (38)
> V[3];
                                 [80, 81]                                      (39)
> [V[1][1], V[1][2], V[2][1], V[2,2]];
                               [43, 9, 79, 2]                                  (40)
> ListToString(%);
                                  "I'm "                                       (41)
> L:=[];
  for j from 1 to nops(V) do
    for i from 1 to 2 do
      L:=[ op(L), V[j][i] ];
    od;
  od;
                                  L := [ ]                                     (42)
> L;
        [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86, 81, 84, 3, 58]    (43)
> V;
     [[43, 9], [79, 2], [80, 81], [86, 2], [67, 2], [88, 71], [69, 86], [81, 84], [3, 58]]   (44)
> M:=[];
  for j from 1 to nops(V) do
      M:=[ op(M), op(V[j]) ];
  od;
                                  M := [ ]
                                M := [43, 9]
                             M := [43, 9, 79, 2]
                         M := [43, 9, 79, 2, 80, 81]
                      M := [43, 9, 79, 2, 80, 81, 86, 2]
                   M := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2]
               M := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71]
           M := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86]
       M := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86, 81, 84]
    M := [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86, 81, 84, 3, 58]   (45)
> map(sin,[0, Pi/2, 27, cat]);
                        [0, 1, sin(27), sin(cat)]                              (46)
> map(op,V);
        [43, 9, 79, 2, 80, 81, 86, 2, 67, 2, 88, 71, 69, 86, 81, 84, 3, 58]    (47)
```

```
> VectToString:=proc( vlist )
    ListToString(map(op,vlist))
  end:
> VectToString(V);
                         "I'm not a vector!X"                                    (48)
```