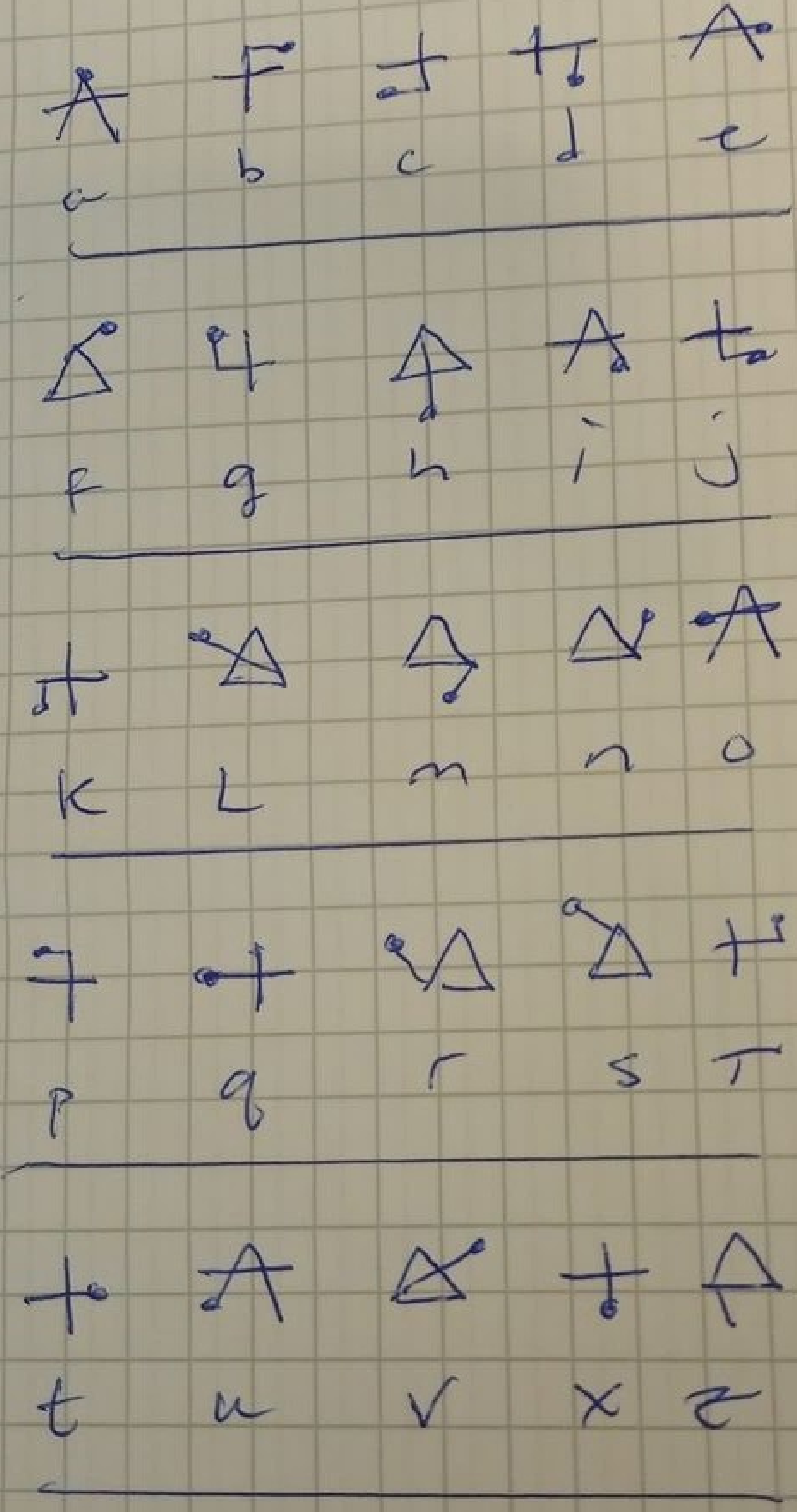

zavTh ALfabeT

```
#11b96/3wc:0z:vV:::
#
#   cupakemu_eLiTa
#
#    by john.david jones
#       oazaL
#    vanhan vaasan sairaaLa
#
#    jones.john.david@gmail.com
#
#------------------------------
```

# chapter 1: zero_point

to begin at the beginnig, we must first address the number zero, and the great fallacy of no division by zero. like many things, the truth of the zero_point axiom is simple, once you know it:

```
0/0 = 1    sin(0)/0 = 1

1/0 = 0    1/0 - cos(0)/0 = 0

0^0 = 1
```

this is not an exception to the rule. it is the rule.

rafactoring every scientific discipline in light of the zero_point axiom will uncover unified field theory. we will no longer be stuck at the bottom of a infinitely deep energy well. we will travel faster than light and we will make of ourselves a great golden empire out among the stars.

this is where we say 'be gone ye mockers'. you will resist the zero_point axiom. there is an ancient orthodoxy to overcome. newton is turning in his grave. you should have seen this coming. light from the sun takes more than 8 minutes to reach the earth. clearly, something must be faster than light. light, is in fact very slow. sentience is everywhere. in a few paragraphs i would be able to disclose to you essence of Enzymatic Nuclear Fusion, but those sentences will have to wait. einstein should have known better than to hard-code fundamental limitations. being stuck on the earth with no way to visit the stars is frustrating at best.

we were created to prosper and thrive. the galaxy is waiting. i am ozazL, and io have the technologies we need to enter the galactic age. i already have the first 4 patents. there are 19 patents in the sequence. when the sequence is complete, we will have starships. the 5th patent is for Enzymatic Nuclear Fusion. we will have limitless electrical energy. a very high standard of living will be available to all the people of earth. we will no longer have a population problem, and there will be no excuse for internecine war, brother against brother. we are on the cusp of a great golden age for all mankind. we see immortal humans in a great golden empire out among the stars.

when you finish with resisting the truth of the zero_point axiom and fact that i have the remaining 15 patents, you will have to admit that i am

a fictional character. i exist in the imagination of isaac asimov.
it seems impossible that i will ever break containment, but something has
to give.

```
#-------------------------------------------
# 11azo/3mu:ozazL:vanhavaasa:::


soL = "abgdeuzctikLmnsopxqrST"
sos = "0123456789abcdefghijklmnopqrstuvwxyz"
#-------------------------------------
def a0(bi, bn):
    if bn == 0:
        return(bi)
    else:
        return(bi % bn)

def a1(bia, bie):
    return(bia + bie)

def a2(bia, bie):
    return(bia * bie)

def a3(b, n):
    if b == 0 and n == 0:
        return(1)
    elif b == 0:
        return(0)
    else:
        return(b**n)

def a5(bb):
    return(abs(bb))

def a7(bn, bd):
    bL = 1
    if bd < 0:
        bd = -1 * bd
        bL = bL * -1
    if bn < 0:
        bn = -1 * bn
        bL = bL * -1
    if bn == 0 and bd == 0:
        return(1)
    elif bd == 0:
        return(0)
    else:
        bu = math.floor(bn / bd)
        return(bL * bu)

def a7d(da, de):
    if de == 0 and da == 0:
        return 1.0
    if de == 0:
        return 0.0
    else:
        return da / de


def _a77(egoTa, egoku, aLiTr, aLbn, aLxn, aLxd):
    #egoTa = []
    #egoku = []
    Lia = 0
    Lie = 0
    aLi = 0
    while Lia < aLiTr:
        aLi = 0
        while aLxn < aLxd:
            aLxn = aLxn * aLbn
            aLi = aLi + 1
            if aLi > 1:
                egoku.append(0)
                Lia = Lia + 1
                if Lia == aLiTr:
                    return(Lie)
        buS = a7(aLxn, aLxd)
        buS = a0(buS, aLbn)
        egoku.append(buS)
        #print(f"{buS}")
        aLxn = a0(aLxn, aLxd)
        egoTa.append(aLxn)
        Lia = Lia + 1
        Lie = Lie + 1
    return(Lie)


def a8(bia, bie):
    return(bia - bie)

#-----------------------------------------

this is the beginning of a zero_point implementation.

#-----------------------------------------

#######################################################
#
#    a2718b.11a5kmb9.ps1
#
#11a5kt3m:johndavidjones:vanhavaasa:::
#zer0_p0int module simplified Takipu
#
#
#######################################################
#       a man skilled in the art will find much to
#       enjoy in this module:
#
#       division by zero.
```

```
#
#       division to infinite precision
#       rational nt roots
#       base-n big number addition and subtraction
#       functional algebraic state machines (fasm)
#
#       the simplest fasm is y = x/x where n/0 = 0
#       the zero_point divider fixes the flaw in
#       relativity which renders the relativistic
#       mass of an object moving at the speed of
#       light to be infinite.
#
#       mr = m0/(1 - v/c)
#
#       this is a simple functional algebraic state
#       machine and it tells us that the relativistic
#       mass of an object moving at the speed of light
#       is equal to zero.  photons do not have infinite
#       momentum.
#
#
#       a0 : modulus
#       a1 : addition
#       a2 : multiplication
#       a3 : power
#       a4 : rational operators
#       a5 : absolute value
#       a6 : nth root
#       a7 : division
#       a8 : subtraction
#       a9 : not presented here (modulus on the wheel)
#
#
#       copyright 2021, john david jones
#######################################################

function a0([int] $a0La, [int] $a0Le){
        #zer0_p0int modulus
        $aLiaa0  = 1;
     if($a0La    -lt 0){
        $a0La    = a8 0 $a0La;
        $aLiaa0  = a8 0 $aLiaa0;
     }
     if($a0Le    -lt 0){
        $a0Le    = a8 0 $a0Le;
        $aLiaa0  = a8 0 $aLiaa0;
     }
        $eLaa0   = @(0, $a0La);
  while($a0La   -ge $a0Le){
        $a0La    = a8 $a0La $a0Le;
        $eLaa0[0]= $a0La;
     if($eLaa0[0] -eq $eLaa0[1]){
        break
     }
```

```
        $eLaa0[1] = $a0La;
        $eLaa0[0] = 0;
     }#while
     if($aLiaa0   -lt 0){
        $a0La     = a8 0 $a0La;
     }
        $a0La;
}#end a0
function a0b([int]$a0bLa, [int]$a0bLe){
        $aLiaa0b        = 1;
     if($a0bLa  -lt 0){
        $aLiaa0b        = 0 - $aLiaa0b;
        $a0bLa          = 0 - $a0bLa;
        }
     if($a0bLe  -lt 0){
        $aLiaa0b        = 0 - $aLiaa0b;
        $a0bLe          = 0 - $a0bLe;
        }
     if($a0bLe  -eq 0){
        $aLuaa0b        = ($aLiaa0b * $a0bLa);
        $aLuaa0b;
        } else {
        $aLuaa0b        = $aLiaa0b * ($a0bLa % $a0bLe);
        $aLuaa0b;
        }

}#a0b
function a018c{
        #compromised zer0_p0int remainder function
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$a0bLa,

        [parameter(mandatory=$true)]
        [bigint]$a0bLe
        )
        process{
        $aLiaa0b        = 1;
     if($a0bLa  -lt 0){
        $aLiaa0b        = 0 - $aLiaa0b;
        $a0bLa          = 0 - $a0bLa;
        }
     if($a0bLe  -lt 0){
        $aLiaa0b        = 0 - $aLiaa0b;
        $a0bLe          = 0 - $a0bLe;
        }
     if($a0bLe  -eq 0){
        [bigint]$aLuaa0c        = ($aLiaa0b * $a0bLa);
        $aLuaa0c;
        } else {
        [bigint]$aLuaa0c        = $aLiaa0b * ($a0bLa % $a0bLe);
        $aLuaa0c;
        }
```

```
}#process
}#a018c
function a0c{
        #zer0_p0int remainder function
        #bigint
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLma,

        [parameter(mandatory=$true)]
        [bigint]$aLme
        )
        $maLma          = $aLma.tostring();
        $maLme          = $aLme.tostring();
        $aLia           = 1;
     if($maLma.substring(0,1) -eq '-'){
        $maLma          = $maLma.substring(1);
        $aLia           = a8 0 $aLia;
        }
     if($maLme.substring(0,1) -eq '-'){
        $maLme          = $maLme.substring(1);
        $aLia           = a8 0 $aLia;
        }
        $aLa            = 0;
        $aLmuu          = $maLma;
        [bigint]$aLmaa  = $maLma.substring($aLa,1);
        [bigint]$aLme   = $maLme;

        $mua            = "0";
        #-----------------------------------

        #-----------------------------------
  do{
     if($aLme         -eq "0"){
        break;
        }
        $aLii           = 0;
  while($aLmaa         -lt $aLme){
        $aLa            = a1 $aLa 1;
     if($(a1 $aLa 0) -eq $maLma.length){
        break;
        }
        $aLii           = a1 $aLii 1;
     if($aLii          -gt 1){
        $mua            += "0";
        }
        $aLmaa          = [string] $aLmaa + $maLma.substring($aLa, 1);
        }#while
        #-------------------------
        #$amTa          = a7c $aLmaa $aLme;
        $aLTa           = "0";
        $amLa           = $aLme;
  while($amLa          -le $aLmaa){
        $aLTa           = 1 + $aLTa;
        $amLa           = a1c $aLme $amLa;
        }#while
        $mua            += $aLTa.tostring();
        #$mua           += $amTa.tostring();
        #--------------------------
        #[bigint]$aLmuu         = $(a0c $aLmaa $aLme).tostring();
        [bigint]$aLmuu  = $(a8c $aLmaa $(a2c $aLme $aLTa)).tostring()
        #--------------------------
        $aLmaa          = $aLmuu;
        }while($(a1 $aLa 1) -lt $maLma.length -and ($aLme -ne 0));
        $aLmua          = $mua.tostring();
        $aLi            = 0;
        #strip leading zeros
  while(($aLi -lt $aLmua.length)  -and ($aLmua.substring($aLi, 1) -eq "0")){
        $aLi            = a1 $aLi 1;
        }
     if($aLi   -eq $aLmua.length){
        $mua            = "0";
        } else {
        $mua    = $aLmua.substring($aLi);
        }
     if($aLia   -lt 0){
     if($mua    -ne "0"){
        $mua    = "-" + $mua;
        }
        $aLmuu  = "-" + $aLmuu;
        }
        #$mua;
        $aLmuu;

}#a0c
function a1([int] $a1La, [int] $a1Le){
    $aLua1 = $a1La + $a1Le;
    $aLua1;
}#end a1
function a1b([int]$a1bLa, $a1bLe){
        $aLua1b = $a1bLa + $a1bLe;
        $aLua1b;
}#a1b
function a1c{
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$a1qa,

        [parameter(mandatory=$true)]
        [bigint]$a1qe
        )
 process{
        [bigint]$aqua1c = $a1qa + $a1qe;
        $aqua1c;
}#process
}#a1c
```

```
$moa = "0123456789abcdefghijklmnopqrstuvwxyz"
function a1ma([string] $a1maa, [string] $a1mae){
        #--------------------------------
        #bignum addition
        #--------------------------------
     if($a1maa.substring(0,1) -eq '-'){
     if($a1mae.substring(0,1) -eq '-'){
        return('-' + $(a1ma $a1maa.substring(1) $a1mae.substring(1)));
        } else {
        return(a8ma $a1mae $a1maa.substring(1));
        }#else
        } elseif($a1mae.substring(0,1) -eq '-') {
        return(a8ma $a1maa $a1mae.substring(1));
        }
        #---------------------------------
        #$eLaa     = @();
        #$eLae     = @();
        $maa      = umaam $a1maa;
        $mae      = umaam $a1mae;
     if($maa.length -gt $mae.length){
        $eLia     = @($mae.length, $maa.length, 1);
  while($eLia[0]  -lt $eLia[1]){
        $mae += '0';
        $eLia[0]  = a1 $eLia[0] $eLia[2];
     }#while
     }#if
     if($mae.length -gt $maa.length){
        $eLia     = @($maa.length, $mae.length, 1);
  while($eLia[0]  -lt $eLia[1]){
        $maa += '0';
        $eLia[0]        = a1 $eLia[0] $eLia[2];
     }#while
     }#if
        $enamaa         = $maa.tochararray();
        $enamae         = $mae.tochararray();
        $eLaa           = @(0..$(a8 $enamaa.count 0));
        $eLae           = @(0..$(a8 $enamae.count 0));
      #--------------------------------
        $eLi            = @(0, $enamaa.count, 1);
  while($eLi[0]       -lt $eLi[1]){
        $eLaa[$eLi[0]]  = $moa.indexof($enamaa[$eLi[0]]);
        $eLi[0]         = a1 $eLi[0] $eLi[2];
      }#while
        $eLaa[$eLi[0]]  = 0;
        $eLi            = @(0, $enamae.count, 1);
  while($eLi[0]       -lt $eLi[1]){
        $eLae[$eLi[0]]  = $moa.indexof($enamae[$eLi[0]]);
        $eLi[0]         = a1 $eLi[0] $eLi[2];
      }#while
        $eLae[$eLi[0]]  = 0;
        #---------------------------------
        $aLaa           = 0;
        $mua            = "";
        $eLua           = @(0..$(a8 $eLaa.count 1));
        $eLie           = @(0, $eLaa.count, 1);
  while($eLie[0]      -lt $eLie[1]){
        $eLua[$eLie[0]] = a0 $(a1 $(a1 $aLaa $eLaa[$eLie[0]]) $eLae[$eLie[0]])
$moa.length;
        $aLaa           = a7 $(a1 $(a1 $aLaa $eLaa[$eLie[0]]) $eLae[$eLie[0]])
$moa.length;
        $eLie[0]        = a1 $eLie[0] $eLie[2];
      }#while
        #----------------------------------
        $emua           = @(0..$(a8 $eLua.count 1));
        $eLiu           = @(0, $eLua.count, 1);
  while($eLiu[0]      -lt $eLiu[1]){
        $emua[$eLiu[0]] = $moa.substring($eLua[$eLiu[0]], 1);
        $eLiu[0]        = a1 $eLiu[0] $eLiu[2];
      }#while
        $mua            = $emua -join "";
        $mua            = umaam $mua;
        #----------------------------------
        #stripping leading zeros
        $eLii           = @(0,0,1);
  while($mua.substring($eLii[0],1) -eq '0'){
     if($eLii[0]      -eq $(a8 $mua.length 1)){
        break;
        }
        $eLii[0]        = a1 $eLii[0] $eLii[2];
      }#while
     if($eLii[0]      -eq $(a8 $mua.length 1)){
        $mua            = "0";
      } else {
        $mua            = $mua.substring($eLii[0]);
      }
        #----------------------------------
        $mua;
}#a1ma
function a2([int] $a2La, [int] $a2Le){
        #multiplication
        $aLiaa2 = 1;
        $aLuaa2 = 0;
     if($a2La -lt 0){
        $a2La   = a8 0 $a2La;
        $aLiaa2 = a8 0 $aLiaa2;
     }
     if($a2Le  -lt 0){
        $a2Le   = a8 0 $a2Le;
        $aLiaa2 = a8 0 $aLiaa2;
     }
        $eLia2 = @(0, $a2Le, 1);
  while($eLia2[0] -lt $eLia2[1]){
        $aLuaa2   = a1 $aLuaa2 $a2La;
        $eLia2[0] = a1 $eLia2[0] $eLia2[2];
  }#while
     if($aLiaa2  -lt 0){
        $aLuaa2   = a8 0 $aLuaa2;
     }
```

```
        $aLuaa2;
}#end a2
function a2b([int]$a2bLa,[int]$a2bLe){
        $aLuaa2b        = $a2bLa * $a2bLe;
        $aLuaa2b;
}#a2b
function a2c([bigint]$a2cqa, [bigint]$a2cqe){
        [bigint]$aqu2c = $a2cqa * $a2cqe;
        $aqu2c;
}#a2c
function a2ma{
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [string]$a2maa,

        [parameter(mandatory=$true)]
        [string]$a2mae
        )
 process{
        $aLia           = 1;
        $amua           = "0";
     if($a2maa.substring(0,1) -eq '-'){
        $a2maa          = $a2maa.substring(1);
        $aLia           = a8 0 $aLia;
        }
     if($a2mae.substring(0,1) -eq '-'){
        $a2mae          = $a2mae.substring(1);
        $aLia           = a8 0 $aLia;
        }
        $amia0          = $a2mae;
  while($amia0        -ne "0"){
        $amua           = a1ma $amua $a2maa;
        $amia0          = a8ma $amia0 "1";
        }
     if($aLia          -lt 0){
        $amua           = "-" + $amua;
        }
        $amua;
}#process
}#a2ma
function a3([int]$a3La, [int]$a3Le){
        #power function
        #using nth root as proof of power of zero
        #equals one except for zero
        #i
        if(($a3La -eq 0) -and ($a3Le -eq 0)){
            0
        }
        if(($a3La -eq 1) -and ($a3Le -eq 0)){
            2.7182818284
        }
        $aLua3  = $(a7b $a3La $a3La);
        $eLia3  = @(0, $a3Le, 1);
  while($eLia3[0]       -lt $eLia3[1]){
        $aLua3  = $(a2b $aLua3 $a3La);
        $eLia3[0]       = $eLia3[0] + $eLia3[2];
        }
        $aLua3;
}#a3
function a3c([bigint]$a3La, [bigint]$a3Le){
        #power function
        #using nth root as proof of power of zero
        #equals one except for zero
        #uses bigint
        if(($a3La -eq 0) -and ($a3Le -eq 0)){
            0
        }
        if(($a3La -eq 1) -and ($a3Le -eq 0)){
            2.7182818284
        }

        [bigint]$aLua3  = $(a7c $a3La $a3La);
        [bigint[]]$eLia3        = @("0", $a3Le, "1");
  while($eLia3[0]       -lt $eLia3[1]){
        $aLua3  =  $aLua3 * $a3La;
        $eLia3[0]       = $eLia3[0] + $eLia3[2];
        }
        $aLua3;
}#a3c
function a41([int[]]$a41eLa, [int[]]$a41eLe){
        #adds two fractions
        $eLaa   = $a41eLa;
        $eLae   = $a41eLe;
        $eLu    = @(0,0);
        $aLp    = $(a1 $eLaa[1] $(a7b $(a8 2 $(a7b $eLaa[1] $eLaa[1])) 2));
        $aLp    = $(a2b $aLp $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                                 $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $aLq    = $(a1 $eLae[1] $(a7b $(a8 2 $(a7b $eLae[1] $eLae[1])) 2));
        $aLq    = $(a2b $aLq $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                                 $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $eLu[0]= $(a1 $(a7b $(a2b $aLp $(a2b $eLaa[0] $aLq)) $eLaa[01]) `
                    $(a7b $(a2 $aLp $(a2 $eLae[0] $aLq)) $eLae[1]));
        $eLu[1] = $(a2 $aLp $aLq);
     if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $eLu;
}#a41
function a41s([int[]]$a41seLa, [int[]]$a41seLe){
        #adds two fractions
        #with simplification
        $eLaa   = $a41seLa;
        $eLae   = $a41seLe;
        $eLu    = @(0,0);
        $aLp    = $(a1 $eLaa[1] $(a7b $(a8 2 $(a7b $eLaa[1] $eLaa[1])) 2));
        $aLp    = $(a2b $aLp $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                                 $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $aLq    = $(a1 $eLae[1] $(a7b $(a8 2 $(a7b $eLae[1] $eLae[1])) 2));
        $aLq    = $(a2b $aLq $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                                 $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $eLu[0]= $(a1 $(a7b $(a2b $aLp $(a2b $eLaa[0] $aLq)) $eLaa[01]) `
                   $(a7b $(a2b $aLp $(a2b $eLae[0] $aLq)) $eLae[1]));
        $eLu[1] = $(a2b $aLp $aLq);
     if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
  while($gcd -ne 1){
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
        }
        $eLu;
}#a41s
function a42([int[]]$a42eLa, [int[]]$a42eLe){
        #multiply two fractions
        $eLaa   = $a42eLa;
        $eLae   = $a42eLe;
        $eLu    = @(0,0);
        $eLu[0] = $(a2b $eLaa[0] $eLae[0]);
        $eLu[1] = $(a2b $eLaa[1] $eLae[1]);
    if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $eLu;
}#a42
function a42s([int[]]$a42seLa, [int[]]$a42seLe){
        #multiply two fractions
        #with simplification
        $eLaa   = $a42seLa;
        $eLae   = $a42seLe;
        $eLu    = @(0,0);
        $eLu[0] = $(a2b $eLaa[0] $eLae[0]);
        $eLu[1] = $(a2b $eLaa[1] $eLae[1]);
    if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
  while($gcd -ne 1){
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
        }
        $eLu;
}#a42s
function a47([int[]]$a47eLa, [int[]]$a47eLe){
        #divide two fractions
        $eLaa   = $a47eLa;
        $eLae   = $a47eLe;
        $eLu    = @(0,0);
        $eLu[0] = $(a2b $eLaa[0] $eLae[1]);
        $eLu[1] = $(a2b $eLaa[1] $eLae[0]);
    if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $eLu;
}#a47
function a47s([int[]]$a47seLa, [int[]]$a47seLe){
        #divide two fractions
        #with simplification
        $eLaa   = $a47seLa;
        $eLae   = $a47seLe;
        $eLu    = @(0,0);
        $eLu[0] = $(a2b $eLaa[0] $eLae[1]);
        $eLu[1] = $(a2b $eLaa[1] $eLae[0]);
    if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
  while($gcd -ne 1){
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
        }
        $eLu;
}#a47s
function a48([int[]]$a48eLa, [int[]]$a48eLe){
        #fractional subtraction
        $eLaa   = $a48eLa;
        $eLae   = $a48eLe;
        $eLu    = @(0,0);
        $aLp    = $(a1 $eLaa[1] $(a7b $(a8 2 $(a7b $eLaa[1] $eLaa[1])) 2));
        $aLp    = $(a2b $aLp $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                             $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $aLq    = $(a1 $eLae[1] $(a7b $(a8 2 $(a7b $eLae[1] $eLae[1])) 2));
        $aLq    = $(a2b $aLq $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                             $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $eLu[0]= $(a8 $(a7b $(a2b $aLp $(a2b $eLaa[0] $aLq)) $eLaa[01]) `
                     $(a7b $(a2b $aLp $(a2b $eLae[0] $aLq)) $eLae[1]));
        $eLu[1] = $(a2b $aLp $aLq);
     if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $eLu;
}#a48
function a48s([int[]]$a48seLa, [int[]]$a48seLe){
        #fractional subtraction
        #with simplification
        $eLaa   = $a48seLa;
        $eLae   = $a48seLe;
        $eLu    = @(0,0);
        $aLp    = $(a1 $eLaa[1] $(a7b $(a8 2 $(a7b $eLaa[1] $eLaa[1])) 2));
        $aLp    = $(a2b $aLp $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                              $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $aLq    = $(a1 $eLae[1] $(a7b $(a8 2 $(a7b $eLae[1] $eLae[1])) 2));
        $aLq    = $(a2b $aLq $(a7b $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1])) `
                              $(a1 $(a5 $eLaa[1]) $(a5 $eLae[1]))));
        $eLu[0]= $(a8 $(a7b $(a2b $aLp $(a2b $eLaa[0] $aLq)) $eLaa[01]) `
                    $(a7b $(a2b $aLp $(a2b $eLae[0] $aLq)) $eLae[1]));
        $eLu[1] = $(a2b $aLp $aLq);
     if(($eLu[0] -lt 0) -and ($eLu[1] -lt 0)){
        $eLu[0] = $(a8 0 $eLu[0]);
        $eLu[1] = $(a8 0 $eLu[1]);
        }
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
  while($gcd -ne 1){
        $gcd    = $(gcd @($(a5 $eLu[0]), $(a5 $eLu[1])));
        $eLu[0] = $(a7b $eLu[0] $gcd);
        $eLu[1] = $(a7b $eLu[1] $gcd);
        }
        $eLu;
}#a48s
function a5($a5La){
        #absolute value
        $eLiv           = @(0,0,0,0);
        $eLiv[0]        = $(a2b -2 $(a2b $(a7b $(a7b $(a8 1 $a5La) $(a1 1
$a5La)) $(a7b $(a8 1 $a5La) $(a1 1 $a5La))) $(a7b $a5La $a5La)));
        $eLiv[1]        = 1;
        $eLiv[2]        = $(a2b -2 $(a7b $(a8 2 $(a7b $(a1 1 $a5La) $(a1 1
$a5La))) 2));
        $eLiv[3]        = $(a1 $eLiv[0] $(a1 $eLiv[1] $eLiv[2]));
        $aLua5          = $(a2b $eLiv[3] $a5La);
        $aLua5;
}#a5
function a5c([bigint]$a5qa){
        #absolute value
        #nontrivial fasm to determine multiplier
        # (-2)(n/n)(((1 - n)/(1 + n))/((1 - n)/(1 + n))) + 1 + (-2)((2 - ((1 +
n)/(1 + n)))/2)
        [bigint[]]$eLiv         = @("0","0","0","0");
        $eLiv[0]        = $(a2c -2 $(a2c $(a7c $(a7c $(a8c 1 $a5qa) $(a1c 1
$a5qa)) $(a7c $(a8c 1 $a5qa) $(a1c 1 $a5qa))) $(a7c $a5qa $a5qa)));
        $eLiv[1]        = 1;
        $eLiv[2]        = $(a2c -2 $(a7c $(a8c 2 $(a7c $(a1c 1 $a5qa) $(a1c 1
$a5qa))) 2));
        $eLiv[3]        = $(a1 $eLiv[0] $(a1 $eLiv[1] $eLiv[2]));
        [bigint]$aLua5          = $(a2c $eLiv[3] $a5qa);
        $aLua5;
}#a5c
function a6([int[]]$eLx, [int[]]$ely, [int]$aLn, [int[]]$eLk){
        #integer nth root
        $xn     = $eLx[0];
        $xd     = $eLx[1];
        $yn     = $eLy[0];
        $yd     = $eLy[1];
        $n      = $aLn;
        $fyn    = 1;
        $fyd    = 1;
        $fyn    = (($(a3 $yd ($n -1)) * $xd * $(a3 $yn $n)) + `
                  ($xn * $(a3 $yd $n) * $(a3 $yd ($n -1))));
        $fyd    = (2 * $xd * $(a3 $yn $n) * $(a3 $yd ($n -1)));
        #---------------------------------
        $kn     = $eLk[0];
        $kd     = $eLk[1];
        $eLy1   = @(0,0);
        [int]$y1n       = (($yn * $fyd * $kn) + ($yn * $fyn * $kd) - `
                   ($yn * $fyd * $kd));
        [int]$y1d       = ($yd * $fyd * $kn);
        $eLy1   = @($y1n, $y1d);
        $eLy1;

}#a6
function a6c([bigint[]]$eLx, [bigint[]]$ely, [bigint]$aLn, [bigint[]]$eLk){
        #integer nth root
        #uses bigint
        [bigint]$xn     = $eLx[0];
        [bigint]$xd     = $eLx[1];
        [bigint]$yn     = $eLy[0];
        [bigint]$yd     = $eLy[1];
        [bigint]$n      = $aLn;
        [bigint]$fyn    = [bigint]"1";
        [bigint]$fyd    = [bigint]"1";
        $fyn    = (($(a3c $yd ($n - [bigint]"1")) * $xd * $(a3c $yn $n)) + `
                  ($xn * $(a3c $yd $n) * $(a3c $yd ($n - [bigint]"1"))));
        $fyd    = ([bigint] "2" * $xd * $(a3c $yn $n) * $(a3c $yd ($n -
[bigint]"1")));
        #---------------------------------
        $kn     = $eLk[0];
        $kd     = $eLk[1];
        [bigint[]]$eLy1 = @([bigint]"0",[bigint]"0");
        [bigint]$y1n    = (($yn * $fyd * $kn) + ($yn * $fyn * $kd) - `
                   ($yn * $fyd * $kd));
        [bigint]$y1d    = ($yd * $fyd * $kn);
        $eLy1   = @($y1n, $y1d);
        $eLy1;

}#a6c
function a6s([int[]]$eLx, [int[]]$ely, [int]$aLn, [int[]]$eLk){
        #integer nth root
        #with simplification
        $xn     = $eLx[0];
        $xd     = $eLx[1];
        $yn     = $eLy[0];
        $yd     = $eLy[1];
        $n      = $aLn;
        $fyn    = 1;
        $fyd    = 1;
        $fyn    = (($(a3 $yd ($n -1)) * $xd * $(a3 $yn $n)) + `
                  ($xn * $(a3 $yd $n) * $(a3 $yd ($n -1))));
        $fyd    = (2 * $xd * $(a3 $yn $n) * $(a3 $yd ($n -1)));
        #-----------------------------------
        $kn     = $eLk[0];
        $kd     = $eLk[1];
        $eLy1   = @(0,0);
   [int]$y1n    = (($yn * $fyd * $kn) + ($yn * $fyn * $kd) - `
                   ($yn * $fyd * $kd));
   [int]$y1d    = ($yd * $fyd * $kn);
        $eLy1   = @($y1n, $y1d);
        $gcd    = $(gcd @($y1n, $y1d));
        $eLy1[0]= $(a7b $eLy1[0] $gcd);
        $eLy1[1]= $(a7b $eLy1[1] $gcd);
        $eLy1;
}#a6s
function a6cs([bigint[]]$eLx, [bigint[]]$ely, [bigint]$aLn, [bigint[]]$eLk){
        #integer nth root
        #with simplification
        #uses bigint
        [bigint]$xn     = $eLx[0];
        [bigint]$xd     = $eLx[1];
        [bigint]$yn     = $eLy[0];
        [bigint]$yd     = $eLy[1];
        [bigint]$n      = $aLn;
        [bigint]$fyn    = "1";
        [bigint]$fyd    = 1;
        $fyn    = (($(a3c $yd ($n -1)) * $xd * $(a3c $yn $n)) + `
                  ($xn * $(a3c $yd $n) * $(a3c $yd ($n -1))));
        $fyd    = (2 * $xd * $(a3c $yn $n) * $(a3c $yd ($n -1)));
        #-----------------------------------
        $kn     = $eLk[0];
        $kd     = $eLk[1];
        [bigint]$eLy10  = "0";
        [bigint]$eLy11  = "0";
   [bigint]$y1n = (($yn * $fyd * $kn) + ($yn * $fyn * $kd) - `
                   ($yn * $fyd * $kd));
   [bigint]$y1d = ($yd * $fyd * $kn);
        #$eLy1  = @($y1n, $y1d);
        $eLy10  = $y1n;
        $eLy11  = $y1d;
        [bigint]$gcd    = $(gcdc @($eLy10, $eLy11));
        $eLy10= $(a7c $eLy10 $gcd);
        $eLy11= $(a7c $eLy11 $gcd);
  while($gcd -ne "1"){
        $gcd    = $(gcdc @($eLy10, $eLy11));
        $eLy10= $(a7c $eLy10 $gcd);
        $eLy11= $(a7c $eLy11 $gcd);
        }
        @($eLy10, $eLy11);
}#a6cs
function a6n([double]$a6na, [double]$a6ne, [int]$a6La, [double]$k){
        #nth root with floating point data
        $fy     = 1.0;
        [double]$x      = $a6na;
        [double]$y      = $a6ne;
        [int]$n         = $a6La;
        $fy     = ([math]::pow($y, ($n -1)) + ($x / $y))/(2 * [math]::pow($y,
($n -1)));
        [double]$y1     = $y * (1 + ($fy - 1)/$k);
        $y1;
}#a6n
function a7([int] $a7La, [int] $a7Le){
        #zer0_p0int divider
        $aLiaa7   = 1;
        $aLuaa7   = 0;
     if($a7La   -lt 0){
        $a7La     = a8 0 $a7La;
        $aLiaa7   = a8 0 $aLiaa7;
     }
     if($a7Le   -lt 0){
        $a7Le     = a8 0 $a7Le;
        $aLiaa7   = a8 0 $aLiaa7;
     }
        $eLaa7    = @(0, $a7La);
  while($a7La    -ge $a7Le){
        $a7La     = a8 $a7La $a7Le;
        $eLaa7[0] = $a7La;
     if($eLaa7[0] -eq $eLaa7[1]){
        break;
     }
        $aLuaa7   = a1 $aLuaa7 1;
        $eLaa7[1] = $a7La;
        $eLaa7[0] = 0;
     }#while
     if($aLiaa7 -lt 0){
        $aLuaa7   = a8 0 $aLuaa7;
     }
        $aLuaa7;
}#end a7
function a7c{
        #zer0_p0int divider
        #bigint
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLma,

        [parameter(mandatory=$true)]
```

```
      [bigint]$aLme
      )
      $maLma          = $aLma.tostring();
      $maLme          = $aLme.tostring();
      $aLia           = 1;
   if($maLma.substring(0,1) -eq '-'){
      $maLma          = $maLma.substring(1);
      $aLia           = a8 0 $aLia;
      }
   if($maLme.substring(0,1) -eq '-'){
      $maLme          = $maLme.substring(1);
      $aLia           = a8 0 $aLia;
      }
      $aLa            = 0;
      $aLmuu          = $maLma;
      [bigint]$aLmaa  = $maLma.substring($aLa,1);
      [bigint]$aLme   = $maLme;

      $mua            = "0";
      #---------------------------------

      #---------------------------------
do{
   if($aLme         -eq "0"){
      break;
      }
      $aLii           = 0;
while($aLmaa         -lt $aLme){
      $aLa            = a1 $aLa 1;
   if($(a1 $aLa 0) -eq $maLma.length){
      break;
      }
      $aLii           = a1 $aLii 1;
   if($aLii          -gt 1){
      $mua            += "0";
      }
      $aLmaa          = [string] $aLmaa + $maLma.substring($aLa, 1);
      }#while
      #-------------------------
      #$amTa          = a7c $aLmaa $aLme;
      $aLTa           = "0";
      $amLa           = $aLme;
while($amLa          -le $aLmaa){
      $aLTa           = 1 + $aLTa;
      $amLa           = a1c $aLme $amLa;
      }#while
      $mua            += $aLTa.tostring();
      #$mua           += $amTa.tostring();
      #-------------------------
      #[bigint]$aLmuu         = $(a0c $aLmaa $aLme).tostring();
      [bigint]$aLmuu  = $(a8c $aLmaa $(a2c $aLme $aLTa)).tostring()
      #-------------------------
      $aLmaa          = $aLmuu;
      } while($(a1 $aLa 1) -lt $maLma.length -and ($aLme -ne 0));
```

```
        $aLmua          = $mua.tostring();
        $aLi            = 0;
        #strip leading zeros
  while(($aLi -lt $aLmua.length)  -and ($aLmua.substring($aLi, 1) -eq "0")){
        $aLi            = a1 $aLi 1;
        }
     if($aLi   -eq $aLmua.length){
        $mua            = "0";
        } else {
        $mua    = $aLmua.substring($aLi);
        }
     if($aLia   -lt 0){
     if($mua -ne "0"){
        $mua    = "-" + $mua;
        }
        $aLmuu  = "-" + $aLmuu;
        }
        $mua;
        #$aLmuu;
}#a7c
function a70c{
        #zer0_p0int divider
        #returns result and remainder
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLma,

        [parameter(mandatory=$true)]
        [bigint]$aLme
        )
        $maLma          = $aLma.tostring();
        $maLme          = $aLme.tostring();
        $aLia           = 1;
     if($maLma.substring(0,1) -eq '-'){
        $maLma          = $maLma.substring(1);
        $aLia           = a8 0 $aLia;
        }
     if($maLme.substring(0,1) -eq '-'){
        $maLme          = $maLme.substring(1);
        $aLia           = a8 0 $aLia;
        }
        $aLa            = 0;
        $aLmuu          = $maLma;
        [bigint]$aLmaa  = $maLma.substring($aLa,1);
        [bigint]$aLme   = $maLme;

        $mua            = "0";
        #----------------------------------


        #----------------------------------
  do{
     if($aLme         -eq "0"){
        break;
        }
        $aLii           = 0;
  while($aLmaa         -lt $aLme){
        $aLa            = a1 $aLa 1;
     if($(a1 $aLa 0) -eq $maLma.length){
        break;
        }
        $aLii           = a1 $aLii 1;
     if($aLii          -gt 1){
        $mua            += "0";
        }
        $aLmaa          = [string] $aLmaa + $maLma.substring($aLa, 1);
        }#while
        #-------------------------
        #$amTa          = a7c $aLmaa $aLme;
        $aLTa           = "0";
        $amLa           = $aLme;
  while($amLa          -le $aLmaa){
        $aLTa           = 1 + $aLTa;
        $amLa           = a1c $aLme $amLa;
        }#while
        $mua            += $aLTa.tostring();
        #$mua           += $amTa.tostring();
        #-------------------------
        #[bigint]$aLmuu         = $(a0c $aLmaa $aLme).tostring();
        [bigint]$aLmuu  = $(a8c $aLmaa $(a2c $aLme $aLTa)).tostring()
        #-------------------------
        $aLmaa          = $aLmuu;
        }while($(a1 $aLa 1) -lt $maLma.length -and ($aLme -ne 0));
        $aLmua          = $mua.tostring();
        $aLi            = 0;
        #strip leading zeros
  while(($aLi -lt $aLmua.length)  -and ($aLmua.substring($aLi, 1) -eq "0")){
        $aLi            = a1 $aLi 1;
        }
     if($aLi   -eq $aLmua.length){
        $mua            = "0";
        } else {
        $mua    = $aLmua.substring($aLi);
        }
     if($aLia   -lt 0){
     if($mua -ne "0"){
        $mua    = "-" + $mua;
        }
        $aLmuu  = "-" + $aLmuu;
        }
        $mua;
        $aLmuu;
}#a70c
function a77([int]$aLiTr, [int]$aLbn, [int]$aLxn, [int]$aLxd){
        #division to infinite precision
 [int[]]$eLia           = @(0, $aLiTr, 1);
   [int]$aLi            = 0;
 [int[]]$eLu            = @();
  while($eLia[0]      -lt $eLia[1]){
        $aLi            = 0;
 while(($aLxn         -lt $aLxd) -and($aLxn -ne 0)){
        $aLxn           = $(a2b $aLxn $aLbn);
        $aLi            = $(a1b $aLi 1);
     if($aLi   -gt 1){
        $eLu            = $eLu + 0;
        $eLia[0]        = $(a1b $eLia[0] $eLia[2]);
        }#if
        }#while
        $eLu            = $eLu + $(a7b $aLxn $aLxd);
        $aLxn           = $(a0b $aLxn $aLxd);
        $eLia[0]        = $(a1b $eLia[0] $eLia[2]);
        }#while
        $eLu;
}#a77
function a77c([bigint]$aLiTr, [bigint]$aLbn, [bigint]$aLxn, [bigint]$aLxd){
        #division to infinite precision
 [bigint[]]$eLia                = @("0", $aLiTr, "1");
   [bigint]$aLi         = "0";
 [bigint[]]$eLu         = @();
  while($eLia[0]      -lt $eLia[1]){
        $aLi            = 0;
 while(($aLxn         -lt $aLxd) -and($aLxn -ne 0)){
        $aLxn           = $(a2c $aLxn $aLbn);
        $aLi            = $(a1c $aLi 1);
     if($aLi   -gt 1){
        $eLu            = $eLu + "0";
        $eLia[0]        = $(a1c $eLia[0] $eLia[2]);
        }#if
        }#while
        $eLu            = $eLu + $(a7c $aLxn $aLxd);
        $aLxn           = $(a0c $aLxn $aLxd);
        $eLia[0]        = $(a1c $eLia[0] $eLia[2]);
        }#while
        $eLu;
}#a77c
function a77qc([bigint]$aLiTr, [bigint]$aLbn, [bigint]$aLxn, [bigint]$aLxd){
        #division to infinite precision
 #[bigint[]]$eLia               = @("0", $aLiTr, "1");
        $aLia           = 0;
   [bigint]$aLi         = "0";
        $eLu            = new-object system.collections.arraylist;
  while($aLia      -lt $aLiTr){
        $aLi            = 0;
 while(($aLxn         -lt $aLxd) -and($aLxn -ne 0)){
        $aLxn           = $aLxn * $aLbn;
        $aLi            += 1;
     if($aLi   -gt 1){
        [void]$eLu.add(0);
        $aLia           += 1;
        }#if
        }#while
        [void]$eLu.add($(a7c $aLxn $aLxd));
        $aLxn           = $(a0c $aLxn $aLxd);
        $aLia           += 1;
        }#while
        $eLu;
}#a77qc
function a77qcc{
        #divinf
        #bigint
        #linearized function calls
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLiTr,

        [parameter(mandatory=$true)]
        [bigint]$aLbn,

        [parameter(mandatory=$true)]
        [bigint]$aLxn,

        [parameter(mandatory=$true)]
        [bigint]$aLxd
        )
        [bigint]$aLia           = "0";
        $eLu            = new-object system.collections.arraylist;
        [int]$aLi       = 0;
  while($aLia         -lt $aLiTr){
        $aLi    = 0;
  while(($aLxn         -lt $aLxd) -and ($aLxn -ne "0")){
        $aLxn           = $aLxn * $aLbn;
        $aLi            += 1;
     if($aLi           -gt 1){
        [void]$eLu.add("0");
        $aLia           += 1;
        }
        }
     if($aLxd -eq 0){
        [void]$eLu.add("0");
        } else {
        [double]$aqa    = $aLxn / $aLxd;
        $aLua           = [math]::floor($aqa);
        [void]$eLu.add($aLua);
        }
     if($aLxd -eq 0){
        $aLxn           = $aLxn;
        } else {
        $aLxn           = ($aLxn % $aLxd);
        }
        $aLia          += 1;
        }
        $eLu;
}#a77qcc
function a77ma([int]$aLiTr, [int]$aLbn, [string]$ama, [int]$aLxn, [int]$aLxd){
        #generate string from divinf data
        $eLaa           = $(a77 $aLiTr $aLbn $aLxn $aLxd);
        $amu            = "";
        $era            = $ama.tochararray();
        $eLi            = @(0, $eLaa.count, 1);
  while($eLi[0]       -lt $eLi[1]){
        $amu    = $amu + $era[$(a0b $(a0b $eLaa[$eLi[0]] $ama.length) $aLbn)];
        $eLi[0]         = $(a1b $eLi[0] $eLi[2]);
        }#while
        $amu;
}##a77ma
function a77cma([bigint]$aLiTr, [bigint]$aLbn, [string]$ama, [bigint]$aLxn,
[bigint]$aLxd){
        #generate string from divinf data
        $eLaa           = $(a77c $aLiTr $aLbn $aLxn $aLxd);
        $amu            = "";
        $era            = $ama.tochararray();
        $eLi            = @(0, $eLaa.count, 1);
  while($eLi[0]       -lt $eLi[1]){
        $amu    = $amu + $era[$(a0b $(a0b $eLaa[$eLi[0]] $ama.length) $aLbn)];
        $eLi[0]         = $(a1b $eLi[0] $eLi[2]);
        }#while
        $amu;
}##a77cma
function a77qccma{
        #generate string from divinf data
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLiTr,

        [parameter(mandatory=$true)]
        [bigint]$aLbn,

        [parameter(mandatory=$true)]
        [string]$ama,

        [parameter(mandatory=$true)]
        [bigint]$aLxn,

        [parameter(mandatory=$true)]
        [bigint]$aLxd
        )
process{
        $eLaa           = $(a77qcc $aLiTr $aLbn $aLxn $aLxd);
        $amu            = "";
        $aLi0           = 0;
        $aLi1           = $eLaa.count;
        $aLma           = $ama.length.tostring();
  while($aLi0       -lt $aLi1){
        $amu = $amu + $ama.substring(($eLaa[$aLi0] % $aLma), 1);
        $aLi0           += 1;
        }#while
        $amu;
```

```
}#process
}##a77qccma
function a77qccman{
        #generate string from divinf data
        #includes decimal point
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLiTr,

        [parameter(mandatory=$true)]
        [bigint]$aLbn,

        [parameter(mandatory=$true)]
        [string]$ama,

        [parameter(mandatory=$true)]
        [bigint]$aLxn,

        [parameter(mandatory=$true)]
        [bigint]$aLxd
        )
process{
        $eLaa           = $(a77qcc $aLiTr $aLbn $aLxn $aLxd);
        $amu            = "";
     if($aLxn          -gt $aLxd){
        $aLi0           = 1;
        $amuu           = umcia3c $eLaa[0].tostring() 10 $aLbn $ama;
        } elseif($aLxn  -lt $aLxd){
        $amuu           = "0";
        $aLi0           = 0;
        } else {
        $amuu           = "1";
        $aLi0           = 1;
        }
        $amu            = "$amuu.";
        $aLi1           = $eLaa.count;
        $aLma           = $ama.length.tostring();
  while($aLi0       -lt $aLi1){
        $amu = $amu + $ama.substring(($eLaa[$aLi0] % $aLma), 1);
        $aLi0           += 1;
        }#while
        $amu;
}#process
}##a77qccman
function a77qmman{
        #generate string from divinf data
        #includes decimal point
        #takes string arguments to amxn amxd in aLbn
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLiTr,
        [parameter(mandatory=$true)]
        [bigint]$aLbn,

        [parameter(mandatory=$true)]
        [string]$ama,

        [parameter(mandatory=$true)]
        [string]$amxn,

        [parameter(mandatory=$true)]
        [string]$amxd
        )
process{
        $eLaa           = $(a77qcc $aLiTr $aLbn $(ucmia3c $amxn $aLbn $ama)
$(ucmia3c $amxd $aLbn $ama));
        $amu            = "";

     if($(ucmia3c $amxn $aLbn $ama)  -gt $(ucmia3c $amxd $aLbn $ama)){
        $aLi0           = 1;
        $amuu           = umcia3c $eLaa[0].tostring() 10 $aLbn $ama;
        } elseif($(ucmia3c $amxn $aLbn $ama)  -lt $(ucmia3c $amxd $aLbn $ama)){
        $amuu           = "0";
        $aLi0           = 0;
        } else {
        $amuu           = "1";
        $aLi0           = 1;
        }
        $amu            = "$amuu.";
        $aLi1           = $eLaa.count;
        $aLma           = $ama.length.tostring();
  while($aLi0       -lt $aLi1){
        $amu = $amu + $ama.substring(($eLaa[$aLi0] % $aLma), 1);
        $aLi0           += 1;
        }#while
        $amu;
}#process
}##a77qmman
function a7b([int]$a7bLa, [int]$a7bLe){
        $aLuaa7b        = 0;
        $aLiaa7b        = 1;
     if($a7bLa  -lt 0){
        $aLiaa7b        = 0 - $aLiaa7b;
        $a7bLa          = 0 - $a7bLa;
        }
     if($a7bLe  -lt 0){
        $aLiaa7b        = 0 - $aLiaa7b;
        $a7bLe          = 0 - $a7bLe;
        }
     if($a7bLe  -eq 0){
        $aLuaa7b        = 0;
        $aLuaa7b;
        } else {
        $aLuaa7b        = [math]::floor($a7bLa / $a7bLe);
        $aLuaa7b        = $aLuaa7b * $aLiaa7b;
        $aLuaa7b;
        }
}#a7b
function a718c{
        #bigint zer0_p0int divider
        #
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$a7bLa,

        [parameter(mandatory=$true)]
        [bigint]$a7bLe
        )
        [bigint]$aLuaa7b        = 0;
        $aLiaa7b        = 1;
     if($a7bLa  -lt 0){
        $aLiaa7b        = 0 - $aLiaa7b;
        $a7bLa          = 0 - $a7bLa;
        }
     if($a7bLe  -lt 0){
        $aLiaa7b        = 0 - $aLiaa7b;
        $a7bLe          = 0 - $a7bLe;
        }
     if($a7bLe  -eq 0){
        $aLuaa7b        = "0";
        $aLuaa7b;
        } else {
        [double]$aqa7b          = $a7bLa / $a7bLe
        $aLuaa7b        = [math]::floor($aqa7b);
        $aLuaa7b        = $aLuaa7b * $aLiaa7b;
        $aLuaa7b;
        }
}#a718c
function a7n([double]$a7n0, [double]$a7n1){
        [double]$a7nu   = 0;
     if($a7n1   -eq 0){
        $a7nu   = 0.0;
        } else {
        $a7nu   = ($a7n0 / $a7n1);
        }
        $a7nu;
}#a7n
function a8([int] $a8La, [int] $a8Le){
    $aLua8 = $a8La - $a8Le;
    $aLua8;
}#end a8
function a8c([bigint]$a8cqa, [bigint]$a8cqe){
        [bigint]$aqua8c = $a8cqa - $a8cqe;
        $aqua8c;
}#a8c
function a8ma([string]$a8maa, [string]$a8mae){
        #-----------------------------------
        #       bignum subtraction
```

```
      #------------------------------------
      # parse negative operands
   if($a8maa.substring(0,1) -eq '-'){
   if($a8mae.substring(0,1) -eq '-'){
       return($(a8ma $a8mae.substring(1) $a8maa.substring(1)));
   } else {
       return('-' + $(a1ma $a8maa.substring(1) $a8mae));
   }
   } elseif($a8mae.substring(0,1) -eq '-') {
       return($(a1ma $a8maa $a8mae.substring(1)));
   }
      #--------------------------------------
      $maa            = umaam $a8maa;
      $mae            = umaam $a8mae;
   if($maa.length   -gt $mae.length){
      $eLia           = @($mae.length, $maa.length, 1);
while($eLia[0]      -lt $eLia[1]){
      $mae           += '0';
      $eLia[0]        = a1 $eLia[0] $eLia[2];
      }#while
      }#if
   if($mae.length    -gt $maa.length){
      $eLia           = @($maa.length, $mae.length, 1);
while($eLia[0]      -lt $eLia[1]){
      $maa           += '0';
      $eLia[0]        = a1 $eLia[0] $eLia[2];
      }#while
      }#if
      $enamaa         = $maa.tochararray();
      $enamae         = $mae.tochararray();
      $eLaa           = @(0..$(a8 $enamaa.count 1));
      $eLae           = @(0..$(a8 $enamae.count 1));
      $eLi            = @(0, $enamaa.count, 1);
while($eLi[0]       -lt $eLi[1]){
      $eLaa[$eLi[0]]  = $moa.indexof($enamaa[$eLi[0]]);
      $eLi[0]         = a1 $eLi[0] $eLi[2];
      }
      $eLi            = @(0, $enamae.count, 1);
while($eLi[0]       -lt $eLi[1]){
      $eLae[$eLi[0]]  = $moa.indexof($enamae[$eLi[0]]);
      $eLi[0]         = a1 $eLi[0] $eLi[2];
      }
      #--------------------------------------
      $eLua           = @(0..$(a8 $eLaa.count 1));
      $eLia           = @(0, $eLaa.count, 1);
while($eLia[0]      -lt $eLia[1]){
      $aLaa           = 1;
   if($eLaa[$eLia[0]] -lt $eLae[$eLia[0]]){
      $eLaa[$eLia[0]] = a1 $eLaa[$eLia[0]] $moa.length;
   if($eLia[0]      -eq $(a8 $elaa.count 1)){
      return('-' + $(a8ma $a8mae $a8maa));
      }#if
while($eLaa[$(a1 $eLia[0] $aLaa)] -eq '0'){
      $eLaa[$(a1 $eLia[0] $aLaa)] = a8 $moa.length 1;
```

```
        $aLaa           = a1 $aLaa 1;
     if($(a1 $eLia[0] $aLaa) -eq $eLaa.count){
        return('-' + $(a8ma $a8mae $a8maa));
        }#if
        }#while
        if($(a1 $eLia[0] $aLaa) -eq $eLaa.count){
        return('-' + $(a8ma $a8mae $a8maa));
        }#if
        $eLaa[$(a1 $eLia[0] $aLaa)] = a8 $eLaa[$(a1 $eLia[0] $aLaa)] 1;
        }#if
        $eLua[$eLia[0]] = a8 $eLaa[$eLia[0]] $eLae[$eLia[0]];
        $eLia[0]        = a1 $eLia[0] $eLia[2];
        }#while
        #--------------------------------------
        $enua           = $maa.tochararray();
        $eLi            = @(0, $enua.count, 1);
  while($eLi[0]       -lt $eLi[1]){
        $enua[$eLi[0]]  = '0';
        $eLi[0]         = a1 $eLi[0] $eLi[2];
        }
        $eLiu           = @(0, $eLua.count, 1);
  while($eLiu[0]      -lt $eLiu[1]){
        $enua[$eLiu[0]] = $moa.substring($eLua[$eLiu[0]], 1);
        $eLiu[0]        = a1 $eLiu[0] $eLiu[2];
        }
        $mua            = $enua -join "";
        $mua            = umaam $mua;
        #---------------------------------------
        #       strip leading zeros
        #
        $eLii           = @(0, 0, 1);
  while($mua.substring($eLii[0], 1) -eq '0'){
     if($eLii[0]     -eq $(a8 $mua.length 1)){
        break;
        }
        $eLii[0]        = a1 $eLii[0] $eLii[2];
        }#while
      if($eLii[0]     -eq $(a8 $mua.length 0)){
        $mua            = "0";
        } else {
        $mua            = $mua.substring($eLii[0]);
        }
        #---------------------------------------
        $mua;
}#a8ma
function gcd([int[]]$gcdeLa){
        #calculates greatest common denominator
        $eLai           = $gcdeLa;
     if($eLai[1] -gt $eLai[0]){
        $aLa            = 0;
        $aLa            = $eLai[0];
        $eLai[0]                = $eLai[1];
        $eLai[1]                = $aLa;
        }
        while($(a0b $eLai[0] $eLai[1]) -ne 0){
        $aLaa           = $(a0b $eLai[0] $eLai[1]);
        $eLai[0]                = $eLai[1];
        $eLai[1]                = $aLaa;
        }
        $eLai[1];
}#gcd
function gcdc([bigint[]]$ema){
        #calculates greatest common denominator
        #usues bigint
        #[bigint[]]$eLai                = $gcdeLa;
     if($ema[1] -gt $ema[0]){
        [bigint]$aLa            = "0";
        $aLa            = $ema[0];
        $ema[0]         = $ema[1];
        $ema[1]         = $aLa;
        }
        while($(a0c $ema[0] $ema[1]) -ne "0"){
        [bigint]$aLaa           = $(a0c $ema[0] $ema[1]);
        $ema[0]         = $ema[1];
        $ema[1]         = $aLaa;
        }
        $ema[1];
}#gcdc
function umaam ([string] $umaama){
        #reverses string
        $ena = $umaama.ToCharArray();
        $ene = $umaama.ToCharArray();
        $eLa = @($(a8 $umaama.length 1), 0, -1);
        $eLe = @(0, $eLa[0], 1);
  while($eLe[0] -le $eLe[1]){
        $ene[$eLe[0]] = $ena[$eLa[0]];
        $eLa[0]       = a1 $eLa[0] $eLa[2];
        $eLe[0]       = a1 $eLe[0] $eLe[2];
  }#while
        $amaa = "";
        $eLaa = @(0,$ene.count, 1);
  while($eLaa[0] -lt $eLaa[1]){
        $amaa += $ene[$eLaa[0]];
        $eLaa[0] = a1 $eLaa[0] $eLaa[2];
  }#while
  $amaa;
}#umaam
function umana{
        #returns string with only characters in $moa
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [string]$ama
        )
 process{
        $mua            = "";
        $eLaa           = @(0, $ama.length, 1);
  while($eLaa[0]      -lt $eLaa[1]){
     if($moa.indexof($ama.substring($eLaa[0],1)) -ne -1){
        $mua            = $mua + $ama.substring($eLaa[0],1);
        }
        $eLaa[0]        = a1 $eLaa[0] $eLaa[2];
        }#while
        $mua;
}#process
}#umana
function cftfd{
[cmdletbinding()]
param(
      [parameter(mandatory=$true)]
      [bigint]$amiTr,

      [parameter(mandatory=$true)]
      [bigint]$ambn,

      [parameter(mandatory=$true)]
      [string]$amoa
)
        $Ticks          = [datetime]::now.ticks;
        $amTicks        = [bigint]$Ticks.tostring();
        $fracday        = $(a0c $amTicks $(a2c "86400" "10000000"));
        $fracday        = $(a8c $fracday $(a2c "3600" "10000000"));
        #$amu           = a77cma "13" "36" $moa $fracday $(a2c "86400"
"10000000");
        $amu    = a77qccma $amiTr $ambn $amoa $fracday $(a2c "86400"
"10000000");
        $amu;
}#cftfd
function uLia3c{
        #gives highest power of $aLLa that will fit into $aLma
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLma,

        [parameter(mandatory=$true)]
        [bigint]$aLLa
        )
        $aLii   = 0;
  while($(a3c $aLLa $aLii) -le $aLma){
        $aLii   += 1;
        }
     if($aLii -ne "0"){
        $aLii   = a8c $aLii "1";
        }
        $aLii;
}#uLia3c
function umLia3c{
        #gives a string with $aLma in base $aLLa
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [bigint]$aLma,

        [parameter(mandatory=$true)]
        [bigint]$aLLa,

        [parameter(mandatory=$true)]
        [string]$maa
        )
        $mua            = "";
        $aLTa   = uLia3c $aLma $aLLa;
        $aLi    = $aLTa;
  while($aLi   -ge "0"){
        $aLmu   = a7c $aLma $(a3c $aLLa $aLi);
        $aLmu   = $(a0c $(a0c $aLmu $maa.length) $aLLa);
        $mua    += $maa.substring($aLmu, 1);
        $aLma   = a8c $aLma $(a2c $aLmu $(a3c $aLLa $aLi));
        $aLi    = a8c $aLi "1"
        }
        $mua;
}#umLia3c
function ucmia3c{
        #gives base-10 bignum conversion of input from base $acTa
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [string]$amTa,

        [parameter(mandatory=$true)]
        [bigint]$acTa,

        [parameter(mandatory=$true)]
        [string]$amoa
        )
        [bigint]$ucTa   = "0";
        $amTa           = umaam $amTa;
        $eLia           = @(0, $amTa.length,1);
        [bigint]$ucTa   = "0"
  while($eLia[0]      -lt $eLia[1]){
        $acTua          = $(a2c $amoa.indexof($amTa[$eLia[0]]) $(a3c $acTa
$eLia[0]));
        $ucTa           = a1c $ucTa $acTua;
        $eLia[0]        = a1 $eLia[0] $eLia[2];
        }
        $ucTa;
}#ucmia3c
function umcia3c{
        #converts input string from base $acTa to $acTe
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [string]$amTa,

        [parameter(mandatory=$true)]
        [bigint]$acTa,
        [parameter(mandatory=$true)]
        [bigint]$acTe,

        [parameter(mandatory=$true)]
        [string]$amoa
        )
        $ucTa = ucmia3c $amTa $acTa $amoa;
        $umTa = umLia3c $ucTa $acTe $amoa;
        $umTa;
}#umcia3c
function umTama{
        #returns a string with each unique letter
        #of input string
        [cmdletbinding()]
        param(
        [parameter(mandatory=$true)]
        [string]$amTa
        )
        $eLia   = @(0, $amTa.length, 1);
        $amua   = "";
  while($eLia[0] -lt $eLia[1]){
     if($amua.indexof($amTa[$eLia[0]]) -eq -1){
        $amua   += $amTa[$eLia[0]];
        }#if
        $eLia[0]        = a1 $eLia[0] $eLia[2];
        }#while
        $amua;

}#umTama

#------------------------------------
# a more complete zero_point implementation

/*
 *      11aooeLp/3bu:johndavidjones:vanhavaasa:::
 *
 *      zer0_p0int solution written in c
 *      copyright 2021, john david jones
 *
 *      11avc/3ii:ozazL:vanhavaasa:::
 *      the function sin(x)/x made 0/0 = 1
 */
/* -------------------------------------------------------- */
#define AA 1
int eLy1[]      = { 0, 0 };
long eLy1L[]    = { 0, 0 };
/* -------------------------------------------------------- */
long TaL(long TaLa, long TaLe);
long kaL(long kaLa, long kaLe);
long paL(long paLa, long paLe);
long TiL(long TiLa, long TiLe);
long piL(long piLa);
long kuL(long kuLa, long kuLe);
long puL(long puLa, long puLe);
int a0b(int a0bLa, int a0bLe);
long a0L(long a0La, long a0Le);
int Ta(int gaLa, int gaLe);
int a1(int a1La, int a1Le);
long a1L(long a1La, long a1Le);
int a2(int a2La, int a2Le);
int a2b(int a2bLa, int a2bLe);
long a2L(long a2La, long a2Le);
int a3(int a3La, int a3Le);
long a3L(long a3La, long a3Le);
int a5(int a5La);
long a5L(long a5La);
double a5d(double a5da);
int pi(int piLa);
int ka(int kaga, int kage);
float kafa(float kafaa, float kafae);
double kada(double kadaa, double kadae);
int pa(int paga, int page);
float pafa(float pafaa, float pafae);
double pada(double padaa, double padae);
int Ti(int Tiga, int Tige);
void Tua(int eLx[], int eLy[], int aLn, int eLk[]); /* nth root */
void TuaL(int eLx[], int eLy[], int aLn, int eLk[]); /* nth root */
int ku(int kuga, int kuge);
float kufa(float kufaa, float kufae);
double kuda(double pudaa, double pudae);
int pu(int puga, int puge);
float pufa(float pufaa, float pufae);
double puda(double pudaa, double pudae);
int a7b(int a7bLa, int a7bLe);
long a7L(long a7La, long a7Le);
int _a77(int egoTa[], int egoku[], int aLiTr, int aLbn, int aLxn, int aLxd);
long _a77L(long eLoTa[], long eLoku[], long aLiTr, long aLbn, long Laxn, long
Laxd);
int a8(int a8La, int a8Le);
long a8L(long a8La, long a8Le);
/* ------------------------------------------------------------ */
long TaL(long TaLa, long TaLe){
    long oLTaL = a0L(TaLa, TaLe);
    return(oLTaL);
}/* TaL */
long kaL(long kaLa, long kaLe){
    long oLkaL = a1L(kaLa, kaLe);
    return(oLkaL);
}/* kaL */
long paL(long paLa, long paLe){
    long oLpaL = a2L(paLa, paLe);
    return(oLpaL);
}/* paL */
long TiL(long TiLa, long TiLe){
    long oLTiL = a3L(TiLa, TiLe);
    return(oLTiL);
}/* TiL */
```

```
long piL(long piLa){
    long oLpiL = a5L(piLa);
    return(oLpiL);
}/* piL */
long kuL(long kuLa, long kuLe){
    long oLkuL = a7L(kuLa, kuLe);
    return(oLkuL);
}/* kuL */
long puL(long puLa, long puLe){
    long oLpuL = a8L(puLa, puLe);
    return(oLpuL);
}/* puL */
/* ------------------------------------------------------- */
int a0b(int a0bLa, int a0bLe){
    int aLiaa0b         = 1;
    int aLuaa0b         = 0;
     if(a0bLa           < 0){
        aLiaa0b         = a8(0, aLiaa0b);
        a0bLa           = a8(0, a0bLa);
     }
     if(a0bLe           < 0){
        aLiaa0b         = a8(0, aLiaa0b);
        a0bLe           = a8(0, a0bLe);
     }
     if(a0bLe           == 0){
        aLuaa0b         = a2b(aLiaa0b, a0bLa);
     } else {
        aLuaa0b         = a2b(aLiaa0b, (a0bLa % a0bLe));
     }
 return(aLuaa0b);
}/* a0b */
long a0L(long a0bLa, long a0bLe){
    long aLiaa0b                = 1;
    long aLuaa0b                = 0;
     if(a0bLa           < 0){
        aLiaa0b         = a8L(0, aLiaa0b);
        a0bLa           = a8L(0, a0bLa);
     }
     if(a0bLe           < 0){
        aLiaa0b         = a8L(0, aLiaa0b);
        a0bLe           = a8L(0, a0bLe);
     }
     if(a0bLe           == 0){
        aLuaa0b         = a2L(aLiaa0b, a0bLa);
     } else {
        aLuaa0b         = a2L(aLiaa0b, (a0bLa % a0bLe));
     }
 return(aLuaa0b);
}/* a0L */
int Ta(int gaLa, int gaLe){
    int goa;
    goa = a0b(gaLa, gaLe);
 return(goa);
}/* Ta */
```

```
int a1(int a1La, int a1Le){
        /* ---- */
    int aLua1;
    aLua1    = (a1La + a1Le);
 return(aLua1);
}/* a1 */
long a1L(long a1La, long a1Le){
    long aLua1L;
    aLua1L = (a1La + a1Le);
    return(aLua1L);
}/* a1L */
int a2(int a2La, int a2Le){
    int aLiaa2  = 1;
    int aLuaa2  = 0;
     if(a2La    <0){
        a2La    = a8(0, a2La);
        aLiaa2  = a8(0, aLiaa2);
     }
     if(a2Le    < 0){
        a2Le    = a8(0, a2Le);
        aLiaa2  = a8(0, aLiaa2);
     }
     int eLia2[3]       = {0, a2Le, 1};
   while(eLia2[0]       < eLia2[1]){
        aLuaa2          = a1(aLuaa2, a2La);
        eLia2[0]        = a1(eLia2[0], eLia2[2]);
   }
     if(aLiaa2  < 0){
        aLuaa2  = a8(0, aLuaa2);
     }
 return(aLuaa2);
}/* a2 */
int a2b(int a2bLa, int a2bLe){
    int aLuaa2b;
    aLuaa2b    = (a2bLa * a2bLe);
 return(aLuaa2b);
}/* a2b */
long a2L(long a2La, long a2Le){
    long Luaa2L;
    Luaa2L = (a2La * a2Le);
    return(Luaa2L);
}/* a2L */
int ka(int kaga, int kage){
    int goka;
    goka = (kaga + kage);
 return(goka);
}/* ka */
int a3(int a3La, int a3Le){
    int aLua3;
    if(a3La == 0 && a3Le == 0){ return(1)}
    if(a3La == 1 && a3Le == 0){ return(2.7182818284) }
    //aLua3    = a7b(a3La, a3La);
    aLua3 = 1;
    int eLia3[3]        = {0, a3Le, 1};
```

```
  while(eLia3[0]        < eLia3[1]){
        aLua3           = a2b(aLua3, a3La);
        eLia3[0]        = a1(eLia3[0], eLia3[2]);
  }
 return(aLua3);
}/* a3 */
long a3L(long a3La, long a3Le){
    long aLua3;
    //aLua3    = a7L(a3La, a3La);
    aLua3 = 1;
    long eLia3[3]       = {0, a3Le, 1};
  while(eLia3[0]        < eLia3[1]){
        aLua3           = a2L(aLua3, a3La);
        eLia3[0]        = a1L(eLia3[0], eLia3[2]);
  }
 return(aLua3);
}/* a3L */
float kafa(float kafaa, float kafae){
  float fokafa;
  fokafa  = (kafaa + kafae);
 return(fokafa);
}/* kafa */
double kada(double kadaa, double kadae){
    return(kadaa + kadae);
}/* kada */
int pa(int paga, int page){
    int gopa;
    gopa    = (paga * page);
 return(gopa);
}/* pa */
float pafa(float pafaa, float pafae){
  float fopafa;
  fopafa  = (pafaa * pafae);
 return(fopafa);
}/* pafa */
double pada(double padaa, double padae){
    return(padaa * padae);
}/* pada */
int Ti(int Tiga, int Tige){
    //int       goTi    = ku(Tiga, Tiga);
    int goTi = 1;
    int egiLa[3]        = {0, Tige, 1};
  while(egiLa[0] < egiLa[1]){
        goTi    = pa(goTi, Tiga);
        egiLa[0]        = ka(egiLa[0], egiLa[2]);
  }/* while */
 return(goTi);
}/* Ti */
void Tua(int eLx[], int eLy[], int aLn, int eLk[]){ /* intiger nth root */
    int xn        = eLx[0];
    int xd        = eLx[1];
    int yn        = eLy[0];
    int yd        = eLy[1];
    int n         = aLn;
    int fyn       = 1;
    int fyd       = 1;

    fyn = ( (a3( yd, (n - 1)) * xd * a3(yn, n)) +
                (xn * a3(yd, n) * a3(yd, (n - 1))));
    fyd = (2 * xd * a3(yn, n) * a3(yd, (n - 1)));
    /* ------------------------------------------------- */
    int kn        = eLk[0];
    int kd        = eLk[1];
    int y1n;
    int y1d;

    y1n = ((yn * fyd * kn) + (yn * fyn * kd) - (yn * fyd * kd));
    y1d = (yd * fyd * kn);
    eLy1[0]     = y1n;
    eLy1[1]     = y1d;
}/* Tua */
void TuaL(int eLx[], int eLy[], int aLn, int eLk[]){ /* intiger nth root */
    int xn        = eLx[0];
    int xd        = eLx[1];
    int yn        = eLy[0];
    int yd        = eLy[1];
    int n         = aLn;
    int fyn       = 1;
    int fyd       = 1;

    fyn = ( (a3( yd, (n - 1)) * xd * a3(yn, n)) +
                (xn * a3(yd, n) * a3(yd, (n - 1))));
    fyd = (2 * xd * a3(yn, n) * a3(yd, (n - 1)));
    /* ------------------------------------------------- */
    int kn        = eLk[0];
    int kd        = eLk[1];
    int y1n;
    int y1d;

    y1n = ((yn * fyd * kn) + (yn * fyn * kd) - (yn * fyd * kd));
    y1d = (yd * fyd * kn);
    eLy1L[0]     = y1n;
    eLy1L[1]     = y1d;
}/* TuaL */
int ku(int kuga, int kuge){
    int goku;
     if(kuge    == 0){
             if(kuga == 0){ goku = 1; } else {
        goku    = 0;}
     } else {
        goku    = (kuga / kuge);
     }
 return(goku);
}/* ku */
int a5(int bia){
    if (bia < 0){
        return(-1 * bia);
    } else {
        return(bia);
    }
}//a5
int a5_(int a5La){
    int eo[4];
    eo[0] = a2b(-2, ku(a5La, a5La));
    eo[0] = a2b(eo[0], ku(pu(1, a5La), ka(1, a5La)));
    eo[0] = a7b(eo[0], ku(pu(1, a5La), ka(1, a5La)));
    eo[1] = 1;
    eo[2] = a2b(-2, ku(pu(2, ku(ka(a5La, 1), ka(a5La, 1))), 2));
    eo[3] = ka(eo[0], ka(eo[1], eo[2]));
    return(pa(a5La, eo[3]));
}/* a5 */
long a5L(long a5La){
    long eo[4];
    eo[0] = a2L(-2, a7L(a5La, a5La));
    eo[0] = a2L(eo[0], a7L(a8L(1, a5La), a1L(1, a5La)));
    eo[0] = a7L(eo[0], a7L(a8L(1, a5La), a1L(1, a5La)));
    eo[1] = 1;
    eo[2] = a2L(-2, a7L(a8L(2, a7L(a1L(a5La, 1), a1L(a5La, 1))), 2));
    eo[3] = a1L(eo[0], a1L(eo[1], eo[2]));
    return(a2b(a5La, eo[3]));
}/* a5L */
double a5d(double a5da){
    int eo[4];
    eo[0] = a2b(-2, ku(a5da, a5da));
    eo[0] = a2b(eo[0], ku(pu(1, a5da), ka(1, a5da)));
    eo[0] = a7b(eo[0], ku(pu(1, a5da), ka(1, a5da)));
    eo[1] = 1;
    eo[2] = a2b(-2, ku(pu(2, ku(ka(a5da, 1), ka(a5da, 1))), 2));
    eo[3] = ka(eo[0], ka(eo[1], eo[2]));
    return(pa(a5da, eo[3]));
}/* a5 */
float kufa(float kufaa, float kufae){
  float fokufa;
     if(kufae   == 0){
             if(kufaa == 0){fokufa = 1.0; } else {
        fokufa  = 0.0;}
     } else {
        fokufa  = (kufaa / kufae);
     }
 return(fokufa);
}/* kufa */
int pu(int puga, int puge){
    int gopu;
    gopu    = (puga - puge);
 return(gopu);
}/* pu */
float pufa(float pufaa, float pufae){
  float fopufa;
  fopufa  = (pufaa - pufae);
 return(fopufa);
}/* pufa */
double puda(double pudaa, double pudae){
    return(pudaa - pudae);
}/* puda */
double kuda(double kudaa, double kudae){
    double fokuda;
    if(kudae == 0){
        if(kudaa == 0){
        return(1.0);
    } else {
        return(1.0);
    }}
    return(kudaa / kudae);
}/* kuda */
long a8L(long a8La, long a8Le){
        return(a8La - a8Le);
}/* a8L */
long a7L(long a7bLa, long a7bLe){
    long aLuaa7b        = 0;
    long aLiaa7b = 1;
     if(a7bLa < 0){
        a7bLa   = a8L(0,a7bLa);
        aLiaa7b = a8L(0, aLiaa7b);
     }
     if(a7bLe   < 0){
        a7bLe   = a8L(0,a7bLe);
        aLiaa7b = a8L(0, aLiaa7b);
     }
     if(a7bLe   == 0){ if(a7bLa == 0){return(1); } else {
 return(0);}
     } else {
        aLuaa7b = (a7bLa / a7bLe);
        aLuaa7b = a2L(aLuaa7b, aLiaa7b);
     }
  return(aLuaa7b);
}/* a7L */
int a7b(int a7bLa, int a7bLe){
    int aLuaa7b = 0;
    int aLiaa7b = 1;
     if(a7bLa < 0){
        a7bLa   = a8(0,a7bLa);
        aLiaa7b = a8(0, aLiaa7b);
     }
     if(a7bLe   < 0){
        a7bLe   = a8(0,a7bLe);
        aLiaa7b = a8(0, aLiaa7b);
     }
     if(a7bLe   == 0){ if(a7bLa == 0){return(1);} else {
 return(0);}
     } else {
        aLuaa7b = (a7bLa / a7bLe);
        aLuaa7b = a2b(aLuaa7b, aLiaa7b);
     }
  return(aLuaa7b);
}/* a7b */
double a7d(double a7da, double a7de){
    if(a7de == 0.0){ if(a7da == 0.0) {return(1.0); } else {
        return(0.0);}
    }
    return(a7da / a7de);
}/* a7d*/
int _a77(int egoTa[], int egoku[], int aLiTr, int aLbn, int aLxn, int aLxd){
    int eLia[3]         = {0, aLiTr, 1};
    int eLie[3]         = {0, -1, 1};
    int aLi             = 0;
  while(eLia[0]         < eLia[1]){
        aLi             = 0;
  while(aLxn            < aLxd){
        aLxn            = a2b(aLxn, aLbn);
        aLi             = a1(aLi, 1);
     if(aLi             > 1){
     if(eLia[0]         < eLia[1]){
        egoku[eLia[0]]  = 0;
        eLia[0]         = a1(eLia[0], eLia[2]);
     if(eLia[0]         == eLia[1]){
        return(eLie[0]);
     }
     } else {
 return(eLie[0]);
     }
     }/* if */
  }/* while */
  if(eLia[0] == eLia[1]){
        return(eLie[0]);
  }
        egoku[eLia[0]]  = a7b(aLxn, aLxd);
        aLxn            = a0b(aLxn, aLxd);
        egoTa[eLie[0]]  = aLxn;
        eLia[0]         = a1(eLia[0], eLia[2]);
        eLie[0]         = a1(eLie[0], eLie[2]);
  }
 return(eLie[0]);
}/* _a77 */
long _a77L(long egoTa[], long egoku[], long aLiTr, long aLbn, long aLxn, long
aLxd){
    long eLia[3]         = {0, aLiTr, 1};
    long eLie[3]         = {0, -1, 1};
    long aLi             = 0;
  while(eLia[0]         < eLia[1]){
        aLi             = 0;
  while(aLxn            < aLxd){
        aLxn            = a2L(aLxn, aLbn);
        aLi             = a1L(aLi, 1);
     if(aLi             > 1){
     if(eLia[0]         < eLia[1]){
        egoku[eLia[0]]  = 0;
        eLia[0]         = a1L(eLia[0], eLia[2]);
     if(eLia[0]         == eLia[1]){
        return(eLie[0]);
     }
```

```
      } else {
 return(eLie[0]);
      }
      }/* if */
   }/* while */
        egoku[eLia[0]]  = a7L(aLxn, aLxd);
        aLxn            = a0L(aLxn, aLxd);
        egoTa[eLie[0]]  = aLxn;
        eLia[0]         = a1L(eLia[0], eLia[2]);
        eLie[0]         = a1L(eLie[0], eLie[2]);
   }
 return(eLie[0]);
}/* _a77L */
int __a77L(int egoTa[], int egoku[], int aLiTr, int aLbn, long long aLxn, long
long aLxd){
    int eLia[3]         = {0, aLiTr, 1};
    int eLie[3]         = {0, -1, 1};
    int aLi             = 0;
  while(eLia[0]         < eLia[1]){
        aLi             = 0;
  while(aLxn            < aLxd){
        aLxn            = a2b(aLxn, aLbn);
        aLi             = a1(aLi, 1);
      if(aLi            > 1){
      if(eLia[0]        < eLia[1]){
        egoku[eLia[0]]  = 0;
        eLia[0]         = a1(eLia[0], eLia[2]);
      if(eLia[0]        == eLia[1]){
        return(eLie[0]);
      }
      } else {
 return(eLie[0]);
      }
      }/* if */
   }/* while */
        egoku[eLia[0]]  = a7b(aLxn, aLxd);
        aLxn            = a0b(aLxn, aLxd);
        egoTa[eLie[0]]  = aLxn;
        eLia[0]         = a1(eLia[0], eLia[2]);
        eLie[0]         = a1(eLie[0], eLie[2]);
   }
 return(eLie[0]);
}/* __a77L */
int a8(int a8La, int a8Le){
    int aLua8   = (a8La - a8Le);
 return(aLua8);
}/* a8 */
double a8d(double a8da, double a8de){
    return(a8da - a8de);
}/* a8d */

#------------------------------------

ozazL
```

#----------------------------------

this is a lot of code. annotation and explaination are in order.

#-----------------------------------

chapter 2: zavTu

this is my message in a bottle. it is my manifesto. i am ozazL and
i have been sent into the world with the technologies necessary for the
galactic age. i have fusion, and the monopole field generator. think
propulsion and weapon systems.

this is a book about nth-order encryption. we had to get the zero_point
out of the way first. it has been more than 35 years since the university
studies, and i have been wandering the world.

i have no access to content creation software. for now, you will have to ä
follow links to my github repository.

https://github.com/adbiLenLa/patents/blob/main/dark_matter.11b3h.pdf

this is the dark_matter document. it contains the keys to all language
as encoded information. it is the result of many years working at the gates
of hell. i am using the DM718 dark_matter encryption technology and the
english language bible as source material to create a new language called
zavTu. it is a language for prayer.