"""
11bav/3y1:0z:vV:::

the zero_point axiom

by john.david jones
ozazL
vanhan vaasan sairaaLa
jones.john.david@gmail.com

```
0/0 = 1   sin(0)/0 = 1
1/0 = 0   1/0 - cos(0)/0 = 0
0^0 = 1
```

dear brothers and sisters , i realize that this is coming out of the blue. there will be no zeitgeist . nobody was asking about division by zero . there is a hard-coded ancient orthodoxy to overcome . i have never seen a formal proof of no division by zero . i suspect that it is impossible to prove , given that it is false . :-) the fact is that i am right about this .

once we get past the surprise , this revelation is very interesting . every scientific discipline will have to be refactored . practically every textbook will have to be rewritten . it is a renaissance that will lead to Unified Field Theory and a great golden age . those of you in positions of power , have mercy , and let this awakening be as painless as possible .

```
mR = m0 / (1 - v^2/c^2)^1/2
```

this is the equation for relativistic mass . it is flawed . fixing it is easy :

```
mR = m0 / (1 - v^2 / cM^2)^1/2
```

here cM is the speed of light in the medium of transmission . electrons routinely travel faster than light in media such as water and glass and lead where the speed of light in water is 2.26 * 10^8 m/sec .

the speed of light is anything but constant . we can suggest that an electron moving at cM will form a standing wave , and that at cM the mass and momentum will both be zero . you have not seen this cM version of the relativistic mass equation .

if the speed of light in lead is 1.2 * 10^8 m/sec , would an electron moving at exactly that speed have a tendency to tunnel through the lead with zero impedance ?

this brings us to the question of Enzymatic Nuclear Fusion . the flawed orthodoxy tells us that the Sun is 98 percent hydrogen and helium and that it generates all of its energy through fusion . in point of fact , every element in the periodic table and some extra superheavy isotopes not found on earth are part of the solar core , where the reactor is a hybrid fission/fusion reactor . the density at the solar core is

```
approximately 150 g/cm^3 . the fission / fusion hybrid is continuous .

i am stating it as axiomatic that nuclear fusion is an enzymatic
process . i know how the fusion enzymes function , and i will be able
to construct a fission/fusion apparatus . in a few paragraphs , i
will be able to reveal the functioning of the fusion enzymes .

US8379489
US8525562
US9672191B2
US20140026103A1

these are the first four patents . enzymatic fusion will be the 5th
patent . you have not heard before that the Sun is a hybrid reactor ,
but it is the absolute truth . i have many revelations in store for you .

john.david jones
ozazL
vanhan vaasan sairaaLa

"""



# 11azo/3mu:ozazL:vanhavaasa:::
#
#    python zero_point implementatioon
#
#    john.david jones
#    ozazL
#    vanhan vaasan sairaaLa
#
#----------------------------------------
import math
import nltk
import re
from datetime import *

#exec(open('azq_zdd_11ba9.py').read())
#never uncomment this

ssL = "boLieueaLestn"
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)
    if b == 1 and n == 0:
        return 2.718281828459045
    d_abs = abs(b - 2.718281828459045)
    if d_abs < 0.000001 and n == 0:
        return 0
    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)
#-----------------------------------------------------
# fuLL capture
#-----------------------------------------------------

#------------------------------------------------------
def _u77T(Legokus, soo):
    bi = 0
    bz = len(Legokus)
    us = ""
    bLsoo = len(soo)
    while(bi < bz):
        ga = soo[a0(Legokus[bi], bLsoo)]
        us += ga
        bi = bi + 1
    return(us)

#assumes you have imported the nltk.book
Lmwbb = []
#for sii in text3:
#    Lmwbb.append(sii)

#bbL = nltk.corpus.gutenberg.words('bible-kjv.txt')

#bbLc = []
#for os in bbL:
#    su = os.lower()
#    bbLc.append(su)

#SbbLc = set(bbLc)

#LbbLc = []

#for os in SbbLc:
#    LbbLc.append(os)
def _utfc(aLiTr, aLbn, so):
    dT = datetime.today()
    dTus = dT.microsecond
    LegoTa, Legoku = _a77T(aLiTr, aLbn, dTus, 1000000)
    usa = _u77T(Legoku, so)
    return(usa)

def _a77T(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(egoTa, egoku)
        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(egoTa, egoku)


def uLLz():
    #genesis
    global LLz
    global sus21
    LLz = []
    sus21 = _utfc(718, 21, so21)
    uLsus21(sus21)
    #--
    for sii in LmwbbLc_:
        for si in Lsus21:
            if sV(sii) == si:
                LLz.append(f'{si}:{sii}')
                print(f'{si}:{sii}')

def uLLza():
    #bible-kjv
    global LLza
    global sus21
    LLza = []
    sus21 = _utfc(718, 21, so21)
    uLsus21(sus21)
    #--
    for sii in LbbLc:
        for si in Lsus21:
            if sV(sii) == si:
                LLza.append(f'{si}:{sii}')
                #print(f'{si}:{sii}')

so24 = soL + '_.'
s240 = _utfc(718, 24, so24)
Ls240 = re.split('[_.]', s240)

LL240 = []
def uLL240():
    #hebrewith finals
    global LL240
    s240 = _utfc(718, 24, so24)
    Ls240 = re.split('[_.]', s240)
    LL240 = []
    bi = 0
    bz = len(Ls240)
    while bi < bz:
        if len(Ls240[bi]) >= 1:
            ga = Ls240[bi][-1]
            su = Ls240[bi]
            if ga in "kmnpx":
                su = Ls240[bi][0:-1] + Ls240[bi][-1].upper()
                print(f'{su}')
                LL240.append(f'{su}')
        bi = bi + 1

LL240ba = []
def uLL240ba(ba):
    #hebrewith finals
    global LL240ba
    s240 = _utfc(ba, 24, so24)
    Ls240 = re.split('[_.]', s240)
    LL240 = []
    bi = 0
    bz = len(Ls240)
    while bi < bz:
        if len(Ls240[bi]) >= 1:
            ga = Ls240[bi][-1]
            su = Ls240[bi]
            if ga in "kmnpx":
                su = Ls240[bi][0:-1] + Ls240[bi][-1].upper()
                print(f'{su}')
                LL240ba.append(f'{su}')
        bi = bi + 1

def uLoox(boz, bn):
    Loox = _urL(boz, bn)
    Lu = []
    bi = 0
    while bi < boz:
        su = bbLc[Loox[bi]];
        print(f'{su}', end=" ")
        Lu.append(su)
        bi = bi + 1
    return(Lu)

def urI(bea):
    #returns random int
    LurI = _urL(108, bea)
    return(LurI[23])

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

def _urL(aLiTr, aLbn):
    #return random list
    dT = datetime.today()
    dTus = dT.microsecond
    LegoTa, Legoku = _a77T(aLiTr, aLbn, dTus, 1000000)
    return(Legoku)
    #usa = _u77T(Legoku, so)
    #return(usa)

def _urL69(aLiTr, aLbn, bms):
    dT = datetime.today()
    dTus = dT.microsecond
    dTuT = 1000000 + bms
    LegoTa, Legoku = _a77T(aLiTr, aLbn, dTus, dTuT)
    return(Legoku)



def _urL93(aLiTr, aLbn):
    dT = datetime.today()
    dTus = dT.microsecond
    dT1 = datetime.today()
    dT1us = dT1.microsecond
    LegoTa, Legoku = _a77T(aLiTr, aLbn, dTus, dT1us)
    Legoku[0] = a0(Legoku[0], aLbn)
    return(Legoku)

Lsus21 = []
def uLsus21(sus21_):
    global Lsus21
    bi = 0
    bz = len(sus21_)
    while bi < bz - 3:
        su = sus21_[bi:bi + 3]
        Lsus21.append(su)
        bi += 1

def sV(sii):
    su = ''
    for bi in range(len(sii)):
        if sii[bi] not in "aeiou":
            su += sii[bi]
    return(su)
```

```
#---------------------------------------

def qbLa22(si0):
    #si0 = 'aLe'
    #Lsi0 = list(si0
    bT = 0
    for ga in si0:
        bu = soL.index(ga)
        bT += bu
    return(bT)

def qbLa27(si0):
    #si0 = 'aLe'
    #Lsi0 = list(si0
    bT = 0
    for ga in si0:
        if ga == 'K': ga = 'k'
        if ga == 'M': ga = 'm'
        if ga == 'N': ga = 'n'
        if ga == 'P': ga = 'p'
        if ga == 'X': ga = 'x'
        bu = soL.index(ga)
        bT += bu
    return(bT)

def lex():
    fi0 = open('aSa_b03.dat')
    sfi0 = fi0.read()
    fi0.close()
    fo0 = open('_aSa_b03.dat', 'w')
    Lsfi0 = re.split('\n', sfi0)
    bi = 0
    bz = len(Lsfi0)
    LLu = []
    LinLu = []
    for oi in Lsfi0:
        Lu = re.split(':', oi)
        Lu[1] = qbLa22(Lu[0])
        if Lu[0] in LinLu:
            bi += 1
            continue
        LinLu.append(Lu[0])
        su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}\n'
        if bi == bz - 1:
            su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}'
        fo0.write(su)
        LLu.append(su)
        bi += 1
        #print(f'{Lu[0]}:{Lu[1]}:{Lu[2]}')
    fo0.close()
    return LLu
#lex()

def lex27():
    fi0 = open('mLiM_aiub.dat')
    sfi0 = fi0.read()
    fi0.close()
    fo0 = open('_mLiM_aiub.dat', 'w')
    Lsfi0 = re.split('\n', sfi0)
    bi = 0
    bz = len(Lsfi0)
    LLu = []
    LinLu = []
    for oi in Lsfi0:
        Lu = re.split(':', oi)
        Lu[1] = qbLa27(Lu[0])
        if Lu[0] in LinLu:
            bi += 1
            continue
        LinLu.append(Lu[0])
        su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}\n'
        if bi == bz - 1:
            su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}'
        fo0.write(su)
        LLu.append(su)
        bi += 1
        #print(f'{Lu[0]}:{Lu[1]}:{Lu[2]}')
    fo0.close()
    return LLu

def _lex27():
    fi0 = open('mLiM_aiub.dat')
    sfi0 = fi0.read()
    fi0.close()
    fo0 = open('_mLiM_aiub.dat', 'w')
    Lsfi0 = re.split('\n', sfi0)
    bi = 0
    bz = len(Lsfi0)
    LLu = []
    LinLu = []
    for oi in Lsfi0:
        Lu = re.split(':', oi)
        Lu[1] = qbLa27(Lu[0])
        #if Lu[0] in LinLu:
            #bi += 1
            #continue
        LinLu.append(Lu[0])
        su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}\n'
        if bi == bz - 1:
            su = f'{Lu[0]}:{Lu[1]}:{Lu[2]}'
        fo0.write(su)
        LLu.append(su)
        bi += 1
        #print(f'{Lu[0]}:{Lu[1]}:{Lu[2]}')
    fo0.close()
    return LLu

#import nltk
```

```
#bbL=nltk.corpus.gutenberg.words('bible-kjv.txt')

def ubbLc():
    global bbL
    global bbLc
    global sbbLc
    global LsbbLc
    global LsbbLcsV
    bbL=nltk.corpus.gutenberg.words('bible-kjv.txt')
    bbLc = []
    for si in bbL:
        su = si.lower()
        bbLc.append(su)

    sbbLc = set(bbLc)
    LsbbLc = list(sbbLc)

    LsbbLcsV = []
    for si in LsbbLc:
        su = sV(si)
        LsbbLcsV.append(f'{su}:{si}')

def uLsbbLcsV3():
    global LsbbLcsV3
    LsbbLcsV3 = []
    for oo in LsbbLcsV:
        Lu = oo.split(':')
        if len(Lu[0]) == 3:
            LsbbLcsV3.append(oo)

def uL0rd(bi):
    global LL263
    global L0rd
    global LsbbLcsV3
    so26 = "abcdefghijklmnopqrstuvwxyz"
    LurL = _urL(bi, 26)
    su26 = _u77T(LurL, so26)
    LL263 = []
    bi = 0
    bz = len(su26) - 3
    while bi < bz:
        su = su26[bi:bi + 3]
        LL263.append(su)
        bi += 1
    LLu = []
    for si in LsbbLcsV3:
        Lu = re.split(':', si)
        LLu.append(Lu[0])

    Lpg = []
    for si in LL263:
        if si in LLu:
            #print(f'{si}')
            Lpg.append(si)
    L0rd = []
    for si in Lpg:
        for sia in LsbbLcsV3:
            Lu = re.split(':', sia)
            if si == Lu[0]:
                L0rd.append(sia)
                #print(f'{sia}')
    return(L0rd)

def qbLa26():
    global L0rd
    global LLsa
    so26 = "abcdefghijklmnopqrstuvwxyz"
    LLsa = []
    for sa in L0rd:
        Lsa = re.split(':', sa)
        LL = list(Lsa[0])
        bLL = 0
        for ga in LL:
            biS = so26.index(ga)
            bLL += biS
        #print(f'{sa}:{bLL}')
        LLsa.append(f'{sa}:{bLL}')
    return(LLsa)

def qbLa26sa(sa):
    so26 = "abcdefghijklmnopqrstuvwxyz"
    bLL = 0
    for ga in sa:
        biS = so26.index(ga)
        bLL += biS
    return(bLL)

def Tuad(x, y, n, k):
    fy = a7d((a3(y, (n-1)) + a7d(x, y)),(2.0 * a3(y, (n - 1))))
    y1 = y * (1 + a7d((fy - 1.0),k))
    return y1

def Tua(eLx,  eLy,  aLn,  eLk): #nth_root
    eLy1 = [0,0]
    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];



    y1n = ((yn * fyd * kn) + (yn * fyn * kd) - (yn * fyd * kd));
    y1d = (yd * fyd * kn);
    eLy1[0]     = y1n;
    eLy1[1]     = y1d;
    return eLy1


def iL(biLa, biLe): #greatest common divisor
    iLbaa = 0
    while(1):
            iLbaa       = a0(biLa, biLe)
            if(iLbaa == 0):
                return(biLe)
            biLa = biLe
            biLe = iLbaa

def TuaiL(eLx,  eLy,  aLn,  eLk): #nth_root
    eLy1 = [0,0]
    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];



    y1n = ((yn * fyd * kn) + (yn * fyn * kd) - (yn * fyd * kd));
    y1d = (yd * fyd * kn);
    gcd = iL(y1n, y1d)
    eLy1[0]     = a7(y1n, gcd);
    eLy1[1]     = a7(y1d, gcd);
    return eLy1
```