1er exámen parcial

1. Decimal → Binario

$0.15_{10} = 0.001001100_2$

.15
0.3
0.6
1.2
0.4
0.8
1.6
1.2
0.4
0.8

2. Suma

$$\begin{array}{r} 110110 \\ +\ 100100 \\ \hline 1011010 \end{array}$$

5 Decimal a Binario

$138_{10} = 10001010_2$

÷	%
69	0
34	1
17	0
8	1
4	0
2	0
1	0
0	1

6. $f(x) = e^x$; $x_0 = 0$

$f(x) = e^x$ $f(0) = 1$
$f'(x) = e^x$ $f'(0) = 1$
$f''(x) = e^x$ $f''(0) = 1$
$f'''(x) = e^x$ $f'''(0) = 1$

$$g(x) = \frac{1}{0!}1 + \frac{1}{1!}x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

$$= 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

$$\frac{1}{\sqrt{e}} = \frac{1}{e^{1/2}} = e^{-1/2}$$

$$e^{\frac{-1}{2}} \approx g\left(\frac{-1}{2}\right) = 1 - \frac{1}{2} + \frac{\left(\frac{-1}{2}\right)^2}{2} + \frac{\left(\frac{-1}{2}\right)^3}{6}$$

$$= \frac{1}{2} + \frac{\frac{1}{4}}{\frac{2}{1}} - \frac{\frac{1}{8}}{\frac{6}{1}}$$

$$= \frac{1}{2} + \frac{1}{8} - \frac{1}{48}$$

$$= \frac{24}{48} + \frac{6}{48} - \frac{1}{48}$$

$$= \frac{29}{48}$$

7. Resta C_2

$13_{10} = 1101_2$

$-13 = 00001101\,C_2 = 11110011_2$

$15_{10} = 1111_2$

$$
\begin{array}{rr}
-13+15 = & 11110011 \\
+ & 00001111 \\
\hline
\cancel{X} & 00000010
\end{array}
$$

8. Multiplicacion Binaria

$$
\begin{array}{r}
11100 \\
\times \quad 101 \\
\hline
11100 \\
000000 \\
1110000 \\
\hline
10001100
\end{array}
$$

```
In [11]: BinarySearch(0, 2, 0.1, 50)

        f(x) = x**4 - 2*x**3 - 4*x**2 + 4*x + 4
        [0, 2]

1. P = 1.0      Er = 100.0 %
2. P = 1.5      Er = 33.33333333333333 %
3. P = 1.25     Er = 20.0 %
4. P = 1.375    Er = 9.090909090909092 %
5. P = 1.4375   Er = 4.3478260869565215 %
6. P = 1.40625  Er = 2.2222222222222223 %
7. P = 1.421875 Er = 1.098901098901099 %
8. P = 1.4140625        Er = 0.5524861878453038 %
9. P = 1.41796875       Er = 0.27548209366391185 %
10. P = 1.416015625     Er = 0.13793103448275862 %
11. P = 1.4150390625    Er = 0.06901311249137336 %

Out[11]: 1.4150390625
```

```
In [16]: NewtonRaphson(2, 0.001, 50)

        f(x) = x**3 - 2*x**2 - 5
        f'(x) = 3*x**2 - 4*x

1. P = 3.25000000000000         Er = 0.384615384615385
2. P = 2.81103678929766         Er = 0.156157049375372
3. P = 2.69798950246853         Er = 0.0419005658567970
4. P = 2.69067715286036         Er = 0.00271766146317258
5. P = 2.69064744851762         Er = 1.10398494456116e-5

Out[16]: 2.69064744851762
```

Aproximaciones

Luis Eduardo Robles Jimenez

0224969

Function

In [ ]:
```
expression = "x - 1"
```

Method

In [ ]:
```
NewtonRaphson(0, 0.1, 50)
```

In [ ]:
```
BinarySearch(0, 2, 1, 50)
```

Newton Raphson

In [ ]:
```
def NewtonRaphson(p0, e, n):
    f = parse_expr(expression)
    d = diff(f, x)
    print("\tf(x) =", f, "\n\tf'(x) =", d, "\n")
    for i in range(n):
        p = p0 - N(f.subs(x, p0))/N(d.subs(x, p0))
        error = abs(N((p - p0)/p))
        print(i + 1, ". ", sep = '', end = '')
        print("P =", p, "\tEr =", error)
        if error < e: return p
        p0 = p
    return p
```

Binary Search

In [ ]:
```
def BinarySearch(a, b, e, n):
    f = parse_expr(expression)
    print("\tf(x) = ", f, "\n\t[", a, ", ", b, "]", "\n", sep = "")
    fp0, p0 = N(f.subs(x, a)), a
    for i in range(n):
        p = a + (b - a)/2
        fp = N(f.subs(x, p))
        error = abs((p - p0)/p)*100
        print(i + 1, ". ", sep = '', end = '')
        print("P = ", p, "\tEr = ", error, " %", sep = '')
        if error < e: return p
        if fp * fp0 > 0: a, fp0 = p, fp
        else: b = p
        p0 = p
    return p
```

Run First

In [ ]:
```
from sympy import *
x = symbols("x")
```