In [47]:
```
#MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA
#Pruebas de software y aseguramiento de la calidad
#Dr. Gerardo Padilla Zárate

#Actividad 4.2. Ejercicios de Programación 1
#CARLOS ENRIQUEZ GORGONIO
#A01793102
#20 de febrero de 2024
```

In [ ]:
```
'''Ejercicio 2. Converter
Detalles
Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).
Req 2. The program shall convert the numbers to binary and hexadecimal base. The results shall be print on a screen and on a file named ConvertionResults.txt.All computation MUST b
Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.
Req 4. The name of the program shall beconvertNumbers.py
Req 5. The minimum format to invoke the program shall be as follows:python convertNumbers.py fileWithData.txt
Req 6. The program shall manage files having from hundreds of items to thousands of items.
Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on
Req 8. Be compliant with PEP8
'''
```

In [13]:
```
!pip install pylint
!pip install pylint[spelling]
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pylint in c:\programdata\anaconda3\lib\site-packages (2.16.2)
Requirement already satisfied: platformdirs>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from pylint) (3.10.0)
Requirement already satisfied: astroid<=2.16.0-dev0,>=2.14.2 in c:\programdata\anaconda3\lib\site-packages (from pylint) (2.14.2)
Requirement already satisfied: isort<6,>=4.2.5 in c:\programdata\anaconda3\lib\site-packages (from pylint) (5.9.3)
Requirement already satisfied: mccabe<0.8,>=0.6 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.7.0)
Requirement already satisfied: tomlkit>=0.10.1 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.11.1)
Requirement already satisfied: dill>=0.3.6 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.3.6)
Requirement already satisfied: colorama>=0.4.5 in c:\programdata\anaconda3\lib\site-packages (from pylint) (0.4.6)
Requirement already satisfied: lazy-object-proxy>=1.4.0 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint) (1.6.0)
Requirement already satisfied: wrapt<2,>=1.14 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint) (1.14.1)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pylint[spelling] in c:\programdata\anaconda3\lib\site-packages (2.16.2)
Requirement already satisfied: platformdirs>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (3.10.0)
Requirement already satisfied: astroid<=2.16.0-dev0,>=2.14.2 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (2.14.2)
Requirement already satisfied: isort<6,>=4.2.5 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (5.9.3)
Requirement already satisfied: mccabe<0.8,>=0.6 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.7.0)
Requirement already satisfied: tomlkit>=0.10.1 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.11.1)
Requirement already satisfied: dill>=0.3.6 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.3.6)
Requirement already satisfied: colorama>=0.4.5 in c:\programdata\anaconda3\lib\site-packages (from pylint[spelling]) (0.4.6)
Requirement already satisfied: pyenchant~=3.2 in c:\users\traba\appdata\roaming\python\python311\site-packages (from pylint[spelling]) (3.2.2)
Requirement already satisfied: lazy-object-proxy>=1.4.0 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint[spelling]) (1.6.0)
Requirement already satisfied: wrapt<2,>=1.14 in c:\programdata\anaconda3\lib\site-packages (from astroid<=2.16.0-dev0,>=2.14.2->pylint[spelling]) (1.14.1)
```

In [5]:
```
import sys
import time
```

In [1]:

```
#Creamos una función para convertir valores a binario considerando el siguiente analisis
"""1. Seleccionar un valor decimal
2. Dividirlo entre 2, despues el cociente se divide entre 2 sucesivamente hasta obtener 0. Esto es con division entera (//)
3. Después se obtienen los módulos (%) de las divisiones,que serán valores 0 o 1
4. Se invierte el orden en la cadena de resultadosy se invierte su orden. Con esto ya tenemos el número binario.
5. En caso de ser un número negativo se hace otra inversion y se añade un 1 adicional en la izquierda inicial"""


def conversor_binario(numero, max_bits=9):

    #Si no se reciben numeros, regresamos un valor nulo
    if numero == 0:
        return 0

    #Creamos nuestro contenedor vacio para almacenar la cadena de valores binarios
    cadena_binaria = ""

    #Obtenemos el valor absoluto de cada numero
    absoluto = abs(numero)

    for _ in range(max_bits - 1):
        cadena_binaria = str(absoluto % 2) + cadena_binaria
        #Cosciente
        absoluto //= 2

    #Se tiene que considerar que los valores negativos deben invertir la cadena, y al principio de esta debe ponerse un 1
    if numero < 0:
        #Cremos una cadena que reemplaza de manera invertida 0 y 1
        inversor = ''.join(['1' if bit == '0' else '0' for bit in cadena_binaria])
        carry = 1
        #Cremos un nuevo contenedor para nuestro string binario
        invertido = ""
        for bit in inversor[::-1]:
            current_bit = str((int(bit) + carry) % 2)
            carry = (int(bit) + carry) // 2
            invertido = current_bit + invertido
        cadena_binaria = '1' + invertido

    else:
        indicador = False
        while indicador is False:
            valor_inicial = cadena_binaria[0]
            if valor_inicial == "1":
                indicador = True
            else:
                #Añadimos los valores con el slicing
                cadena_binaria = cadena_binaria[1:]

    return cadena_binaria
```

In [2]:

```python
#Creamos una función para convertir valores a hexadecimal considerando el siguiente analisis
"""1. Seleccionar un valor decima
2. Dividirlo entre 16, despues el cociente se divide entre 16 sucesivamente hasta obtener 0. Esto es con division entera (//)
3. Después se obtienen los módulos (%) de las divisiones,que serán dentro de nuestro diccionario hexadecimal
4. Se invierte el orden en la cadena de resultados y se invierte su orden. Con esto ya tenemos el número binario."""

def conversor_hexa(numero):

    if numero == 0:
        return 0

    if numero < 0:
        diccionario = "0123456789ABCDEF"
        #Creamos nuestro contenedor de cadena hexadecimal
        cadena_hexa = ""
        #El operador Bitwise << es análogo al >> con la diferencia que en este caso el desplazamiento es realizado a la izquierda.0001, <<3, 1000.
        #Recorremos los bits al tratarse de un valor negativo
        numero = (1 << 32) + numero
        while numero > 0:
            #Modulo
            modulo = numero % 16
            cadena_hexa = diccionario[modulo] + cadena_hexa
            #Absoluto
            numero //= 16
        return cadena_hexa
    else:
        diccionario = "0123456789ABCDEF"
        cadena_hexa = ""
        while numero > 0:
            modulo = numero % 16
            cadena_hexa = diccionario[modulo] + cadena_hexa
            numero //= 16
        return cadena_hexa
```

In [19]:
```
def impresora(ruta):

    try:
        #Iniciamos nuestro timer
        timer = time.time()
        with open(ruta, 'r', encoding="utf-8") as archivo:
            # Creamos una lista contenedora de numeros
            lista_numeros = []
            for renglon in archivo:
                lista_numeros.append(renglon.strip())

            arreglo_base = []

            for num in lista_numeros:
                diccionario = {}
                try:
                    entero = int(num)
                    diccionario['DECIMAL'] = entero
                    binary = conversor_binario(entero)
                    diccionario['BINARIO'] = binary
                    hexadecimal = conversor_hexa(entero)
                    diccionario['HEXADECIMAL'] = hexadecimal
                    #arreglo_base = [].append(diccionario)
                    arreglo_base.append(diccionario)

                except ValueError:
                    error_value = "#VALUE!"
                    diccionario['DECIMAL'] = num
                    diccionario['BINARIO'] = error_value
                    diccionario['HEXADECIMAL'] = error_value
                    #arreglo_base = [].append(diccionario)
                    arreglo_base.append(diccionario)

            #Creamos el arreglo final, incluyendo encabezados
            arreglo_final = "INDICE     DECIMAL BINARIO HEXADECIMAL \n"

            for index, diccionario in enumerate(arreglo_base):
                arreglo_final += (
                    f"{index+1} {diccionario['DECIMAL']} "
                    f"{diccionario['BINARIO']} {diccionario['HEXADECIMAL']}\n"
                )

            fin = time.time()
            temporizador_final = (fin - timer) * 1000

            print(arreglo_final)
            print("\n")
            tiempo_total = f"Tiempo de ejecución: {temporizador_final:.6f} milisegudos"
            print(tiempo_total)
            #Creamos el archivo resultante
            with open("ConvertionResults.txt", "w", encoding="utf-8") as file:
                print(arreglo_final, file=file)
                print("\n", file=file)
                print(tiempo_total, file=file)


    except FileNotFoundError:
        print(f"Error: El archivo '{ruta}' No se encuentra.")
```

In [20]:
```
#Para fines de observar resultados invocamos el archivo desde una ruta local, posteriormente queda la opción de invocarlo desde consola
impresora("C:/Users/traba/Downloads/TC1.txt")
#impresora("C:\Users\traba\Downloads\TC1.txt")
```

INDICE DECIMAL BINARIO HEXADECIMAL
1 6980368 10000 6A8310
2 5517055 11111111 542EFF
3 1336159 1011111 14635F
4 6750185 11101001 66FFE9
5 1771937 10100001 1B09A1
6 360952 11111000 581F8
7 5672561 1110001 568E71
8 916583 1100111 DFC67
9 2700138 1101010 29336A
10 9645053 11111101 932BFD
11 1181110 10110110 1205B6
12 1492185 11011001 16C4D9
13 4018595 10100011 3D51A3
14 7654888 11101000 74CDE8
15 7062453 10110101 6BC3B5
16 2478010 10111010 25CFBA
17 6134768 11110000 5D9BF0
18 8420417 1000001 807C41
19 2917489 1110001 2C8471
20 3340773 11100101 32F9E5
21 1115956 110100 110734
22 9172192 11100000 8BF4E0
23 6271996 11111100 5FB3FC
24 8686939 1011011 848D5B
25 50986 101010 C72A
26 9376410 10011010 8F129A
27 5962327 1010111 5AFA57
28 7686891 11101011 754AEB
29 6615183 10001111 64F08F
30 1864844 10001100 1C748C
31 3329962 10101010 32CFAA
32 3942794 10001010 3C298A
33 2614836 110100 27E634
34 7406772 10110100 7104B4
35 2384190 111110 24613E
36 398347 1011 6140B
37 8698503 10000111 84BA87
38 9551696 1010000 91BF50
39 1019556 10100100 F8EA4
40 1677430 1110110 199876
41 3479629 1001101 35184D
42 9309008 1010000 8E0B50
43 5266170 11111010 505AFA
44 4094340 10000100 3E7984
45 1754055 11000111 1AC3C7
46 5861132 1100 596F0C
47 4471329 100001 443A21
48 8826052 11000100 86ACC4
49 7469325 1101 71F90D
50 1973172 10110100 1E1BB4
51 53145 10011001 CF99
52 3897508 10100100 3B78A4
53 7773386 11001010 769CCA
54 6089829 1100101 5CEC65
55 4223424 11000000 4071C0
56 9761752 11011000 94F3D8
57 7930799 10101111 7903AF
58 3597495 10110111 36E4B7
59 9302948 10100100 8DF3A4
60 2288712 1001000 22EC48
61 197187 1000011 30243
62 5266939 11111011 505DFB
63 221545 1101001 36169
64 7957027 100011 796A23
65 3195361 11100001 30C1E1

66 7106269 11011101 6C6EDD
67 9633312 100000 92FE20
68 9713704 101000 943828
69 91925 10101 16715
70 4418686 1111110 436C7E
71 9682250 1001010 93BD4A
72 2583824 10000 276D10
73 4979126 10110110 4BF9B6
74 6280954 11111010 5FD6FA
75 1228610 1000010 12BF42
76 705518 11101110 AC3EE
77 1017653 110101 F8735
78 500098 10000010 7A182
79 7210727 11100111 6E06E7
80 4250898 10010 40DD12
81 4055028 11110100 3DDFF4
82 2754240 11000000 2A06C0
83 452999 10000111 6E987
84 6831458 1100010 683D62
85 207636 10100 32B14
86 7280635 11111011 6F17FB
87 3308937 10001001 327D89
88 4303570 11010010 41AAD2
89 8375055 1111 7FCB0F
90 1457960 101000 163F28
91 5197625 111001 4F4F39
92 3144371 10110011 2FFAB3
93 6674138 11011010 65D6DA
94 2692106 1010 29140A
95 2276769 10100001 22BDA1
96 1940971 11101011 1D9DEB
97 3288264 11001000 322CC8
98 4819803 1011011 498B5B
99 9431005 11011101 8FE7DD
100 3992019 11010011 3CE9D3
101 8184782 11001110 7CE3CE
102 2847975 11100111 2B74E7
103 7891025 1010001 786851
104 900082 11110010 DBBF2
105 1831532 1101100 1BF26C
106 8428253 11011101 809ADD
107 2905752 10011000 2C5698
108 9763121 110001 94F931
109 257890 1100010 3EF62
110 4699761 1110001 47B671
111 359541 1110101 57C75
112 3446150 10000110 349586
113 4246818 100010 40CD22
114 1848840 1000 1C3608
115 938480 11110000 E51F0
116 227465 10001001 37889
117 3923488 100000 3BDE20
118 8915697 11110001 880AF1
119 4309094 1100110 41C066
120 3891251 110011 3B6033
121 8627956 11110100 83A6F4
122 5884613 11000101 59CAC5
123 1659318 10110110 1951B6
124 1834855 1100111 1BFF67
125 3386751 1111111 33AD7F
126 3000166 1100110 2DC766
127 9135626 1010 8B660A
128 4869260 10001100 4A4C8C
129 76550 110 12B06
130 432271 10001111 6988F
131 251028 10010100 3D494

132 7016218 11010 6B0F1A
133 6896099 11100011 6939E3
134 8386350 101110 7FF72E
135 8637147 11011011 83CADB
136 936705 1 E4B01
137 6602175 10111111 64BDBF
138 1429181 10111101 15CEBD
139 8395138 10000010 801982
140 6132809 1001001 5D9449
141 5936917 10101 5A9715
142 2878578 1110010 2BEC72
143 158885 10100101 26CA5
144 2441957 11100101 2542E5
145 5914794 10101010 5A40AA
146 3999272 101000 3D0628
147 3142897 11110001 2FF4F1
148 8151159 1110111 7C6077
149 5147564 10101100 4E8BAC
150 4595374 10101110 461EAE
151 4234951 11000111 409EC7
152 7880605 10011101 783F9D
153 7009921 10000001 6AF681
154 695580 11100 A9D1C
155 7370443 11001011 7076CB
156 7921729 1000001 78E041
157 8419625 101001 807929
158 7024080 11010000 6B2DD0
159 3905988 11000100 3B99C4
160 1767599 10101111 1AF8AF
161 935136 11100000 E44E0
162 635788 10001100 9B38C
163 8807719 100111 866527
164 317375 10111111 4D7BF
165 9975410 1110010 983672
166 2727968 100000 29A020
167 7444399 10101111 7197AF
168 4065675 10001011 3E098B
169 9925720 1011000 977458
170 2293633 10000001 22FF81
171 7734826 101010 76062A
172 1065463 11110111 1041F7
173 1105617 11010001 10DED1
174 5325800 11101000 5143E8
175 3822527 10111111 3A53BF
176 5503858 1110010 53FB72
177 9214055 1100111 8C9867
178 6521769 10101001 6383A9
179 7923796 1010100 78E854
180 5250236 10111100 501CBC
181 1083154 10010 108712
182 472141 1001101 7344D
183 9597454 1110 92720E
184 1581679 1101111 18226F
185 656751 1101111 A056F
186 345464 1111000 54578
187 4281218 10000010 415382
188 6558883 10100011 6414A3
189 3852986 10111010 3ACABA
190 6263187 10010011 5F9193
191 5828308 11010100 58EED4
192 8058535 10100111 7AF6A7
193 9035191 10110111 89DDB7
194 7922103 10110111 78E1B7
195 9366003 11110011 8EE9F3
196 4555717 11000101 4583C5
197 3526753 1100001 35D061

```
198 3176815 1101111 30796F
199 858440 1001000 D1948
200 2250854 1100110 225866



Tiempo de ejecución: 0.999451 milisegudos
```

In [ ]:

```python
if __name__ == "__main__":
    # Verificamos los parametros en la linea de comando, en caso de estar vacía, solicitamos la información
    if len(sys.argv) != 2:
        print("Introduce la ruta como se muestra: python compute_statistics.py P1/TC2.txt")
        sys.exit(1)

    # obtenemos la ruta de los archivos
    ruta_archivo = sys.argv[1]

    # Invocamos nuestra función principal para imprimir las estadisticas
    impresora(ruta_archivo)
```

In [ ]: