# DOCUMENTO TÉCNICO

Nombre:

Javier Mauricio Romero Deaquiz

### Especificación

**Entrada:**

La entrada debe ser dos tuplas

**Salida:**

La salida debe ser una tupla

### Estrategia

Se realiza únicamente una suma de números complejos, utilizando su parte real e imaginaria

def suma(t1, t2):

    parte\_r = t1[0] + t2[0]

    parte\_i = t1[1] + t2[1]

    return (parte\_r, parte\_i)

### Especificación

**Entrada:**

La entrada debe ser dos tuplas

**Salida:**

La salida debe ser una tupla

### Estrategia

Se realiza únicamente una resta de números complejos, utilizando su parte real e imaginaria

def def resta(t1, t2):

    parte\_r = t1[0] - t2[0]

    parte\_i = t1[1] - t2[1]

    return (parte\_r, parte\_i)

### Especificación

**Entrada:**

La entrada debe ser dos tuplas

**Salida:**

La salida debe ser una tupla

### Estrategia

Se realiza únicamente una multiplicación de números complejos, utilizando su parte real e imaginaria

def multiplicacion(t1, t2):

    parte\_r = (t1[0] \* t2[0]) - (t1[1] \* t2[1])

    parte\_i = (t1[0] \* t2[1]) + (t1[1] \* t2[0])

    return (parte\_r, parte\_i)

### Especificación

**Entrada:**

La entrada debe ser dos tuplas

**Salida:**

La salida debe ser una tupla

### Estrategia

Se realiza únicamente una división de números complejos teniendo en cuenta la fórmula para llevar a cabo esta, utilizando su parte real e imaginaria

def division(t1, t2):

    parte\_r = ((t1[0] \* t2[0]) + (t1[1] \* t2[1])) / ((t2[0] \*\* 2) \* (t2[1] \*\* 2))

    parte\_i = ((t1[0] \* t2[0]) - (t1[1] \* t2[1])) / ((t2[0] \*\* 2) \* (t2[1] \*\* 2))

    return (parte\_r, parte\_i)

### Especificación

**Entrada:**

La entrada debe ser una tupla

**Salida:**

La salida debe ser un numero real

### Estrategia

Se saca el módulo de la tupla realizando una formula de pitágoras

def modulo(t):

    return str(((t[0])\*\*2 + (t[1])\*\*2)\*\*(1/2))

### Especificación

**Entrada:**

La entrada debe ser una tupla

**Salida:**

La salida debe ser una tupla

### Estrategia

Se realiza un cambio de signo en la parte imaginaria

def conjugado(t):

    return (t[0],-t[1])

### Especificación

**Entrada:**

La entrada debe ser una tupla

**Salida:**

La salida debe ser un numero imaginario

### Estrategia

Se realiza una operación matemática para poder sacar la fase

def fase(t):

    return math.atan(t[1] / t[0])

### Especificación

**Entrada:**

La entrada debe ser una tupla

**Salida:**

La salida debe ser una tupla

### Estrategia

Con las dos funciones anteriores, definimos la fase como el angulo y el radio como el modulo, de esta manera hacemos unos condicionales para transformar ese angulo

def carte\_polares(t):

    angulo = fase(t)

    angulo = math.degrees(angulo)

    radio = modulo(t)

    if t[0] > 0:

        if t[1] < 0:

            return (radio, 360 + angulo)

        else:

            return (radio, angulo)

    else:

        if t[1] < 0:

            return (radio, 180 + angulo)

        else:

            return (radio, 180 + angulo)

### Especificación

**Entrada:**

La entrada debe ser una tupla

**Salida:**

La salida debe ser una tupla

### Estrategia

Utilizamos dos funciones de la librería math, para poder gracias a estas, sacar la parte real (x) y la parte imaginaria (y) en donde se emplea una forma para poder pasar de polares a cartesianas

def polares\_carte(p):

    x = p[0] \* math.cos(p[1])

    y = p[0] \* math.sin(p[1])

    return (x,y)

def main():

    #PRUEBAS

    print("PRUEBAS")

    #SUMA

    #PRUEBA 1

    print("SUMA")

    print("PRUEBA 1")

    print(suma((8,2), (10,27)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(suma((30,12),(78,1)))

    #RESTA

    print("RESTA")

    #PRUEBA 1

    print("PRUEBA 1")

    print(resta((8,2), (10,27)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(resta((30,12),(78,1)))

    #MULTIPLICACION

    print("MULTIPLICACION")

    #PRUEBA 1

    print("PRUEBA 1")

    print(multiplicacion((8,2), (10,27)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(multiplicacion((30,12),(78,1)))

    #DIVISION

    print("DIVISION")

    #PRUEBA 1

    print("PRUEBA 1")

    print(division((8,2), (10,27)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(division((30,12),(78,1)))

    #MODULO

    print("MODULO")

    #PRUEBA 1

    print("PRUEBA 1")

    print(modulo((10,27)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(modulo((30,12)))

    #CONJUGADO

    print("CONJUGADO")

    #PRUEBA 1

    print("PRUEBA 1")

    print(conjugado((8,2)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(conjugado((78,-1)))

    #FASE

    print("FASE")

    #PRUEBA 1

    print("PRUEBA 1")

    print(fase((8,10)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(fase((30,78)))

    #De Cartesianas a polares

    print("CARTESIANAS A POLARES")

    #PRUEBA 1

    print("PRUEBA 1")

    print(carte\_polares((8,-10)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(carte\_polares((30,78)))

    #DE POLARES A CARTESIANAS

    print("POLARES A CARTESIANAS")

    #PRUEBA 1

    print("PRUEBA 1")

    print(polares\_carte((8,10)))

    #PRUEBA 2

    print("PRUEBA 2")

    print(polares\_carte((30,78)))

main()