Geometría Computacional
Ejercicio Práctico 1: Preliminares de Geometría

Ailyn Rebollar Pérez
ailynrp12@ciencias.unam.mx

1 Objetivo

Se tiene como objetivo que los estudiantes recuerden e implementen operaciones y propiedades importantes de geometría que se estarán utilizando para las futuras prácticas en el lenguaje de programación **python 3**.

2 Implementación

Se proporcionará un esqueleto del archivo **punto.py** y clase **Punto** que deberán implementar el cual está estructurado de la siguiente manera:

2.1 Atributos

- x : Coordenada x del punto actual.
- y : Coordenada y del punto actual.

2.2 Métodos

- **productoCruz(self, q)**
  Método que regresa el producto cruz del punto actual y el punto recibido q.
- **direccionVuelta(self, q, r)**
  Método que indica la dirección de una vuelta entre 3 puntos. Regresa un valor negativo si se dio vuelta a la izquierda (en contra de las manecillas del reloj), un valor positivo si fue a la derecha (a favor de las manecillas del reloj) y 0 si no hubo vuelta (están sobre la misma recta).
- **anguloPolar(self, q)**
  Método que regresa el ángulo polar correspondiente a dos puntos tomando al punto actual como el origen.
- **anguloGrados(self,q)**
  Método que regresa el ángulo en grados correspondiente a dos puntos a patir de su ángulo polar.
- **distancia(self, q)**
  Método que regresa la distancia entre dos puntos.
- **str(self)**
  Método que regresa la representación en cadena de un punto de manera que se vea (x,y).

Nota

Para la implementación de los métodos presentados pueden utilizar la biblioteca `math` para ocupar funciones ya existentes que **NO resuelvan directamente el problema** presentado en cada método.

Además, no olvides **anotar tu nombre completo y número de cuenta** al inicio del archivo en los comentarios.

Nota para el método anguloPolar() y el método anguloGrados()

Recordemos que un ángulo polar cumple con $0 \leq \theta \leq 2\pi$ mientras que un ángulo en grados cumple con $0 \leq \theta \leq 360$.

Sin embargo, en los ejercicios y prácticas sólo consideraremos que los ángulos polares tomarán los valores de $0 \leq \theta \leq \pi$ mientras que los ángulos en grados tomarán los valores $0 \leq \theta \leq 180$.

2.3 Esqueleto del ejercicio

A continuación se presenta el esqueleto del ejercicio práctico de la clase Punto.

Listing 1: Archivo punto.py

```
'''
Nombre Completo:
Numero de Cuenta:
Clase Punto donde como atributos se
tendran las coordenadas x y.
'''
class Punto:
    '''
    Constructor de la clase Punto
    @param x: coordenada x.
    @param y: coordenada y.
    '''
    def __init__(self,x,y):
        # Escribe aqui tu codigo

    '''
    Metodo que regresa el producto cruz del punto actual y el punto recibido q.
    @param q: punto con el que se realizara el producto cruz.
    @return : el producto cruz entre el punto actual y q.
    '''
    def productoCruz(self, p):
        # Escribe aqui tu codigo


    '''
    Metodo que indica la direccion de una vuelta entre 3 puntos.
    @param q: uno de los puntos para conocer la direccion de la vuelta.
    @param r: uno de los puntos para conocer la direccion de la vuelta.
    @return: regresa -1 si se dio vuelta a la izquierda, 1 si fue a la derecha y 0
    si no hubo vuelta (estan sobre la misma recta).
    '''
    def direccionVuelta(self, q, r):
        # Escribe aqui tu codigo


    '''
    Metodo que regresa el angulo polar correspondiente a dos puntos
    tomando al punto actual como el origen.
    @param q: uno de los puntos para conocer el angulo polar.
    @return: el angulo polar correpondiente a dos puntos tomando al
    punto actual como el origen.
    '''
    def anguloPolar(self, q):
```

```
44          # Escribe aqui tu codigo
45
46      '''
47      Metodo que regresa el  ngulo  en grados correspondiente a dos puntos
48      a partir de su angulo polar.
49      @param q: uno de los puntos para conocer el angulo en grados
50      @return: el angulo en grados correspondiente a dos puntos a partir de
51      su angulo polar.
52      '''
53      def anguloGrados(self,q):
54          # Escribe aqui tu c digo
55
56
57      '''
58      Metodo que regresa la distancia entre dos puntos.
59      @param q: uno de los puntos para determinar la distancia.
60      @return: la distancia entre el punto actual y el punto q
61      '''
62      def distancia(self, q):
63          # Escribe aqui tu codigo
64
65      '''
66      Metodo que regresa la representacion en cadena de un punto
67      de manera que se vea (x,y).
68      @return la representacion en cadena de un punto de manera
69      que se vea (x,y)
70      '''
71      def __str__(self):
72          return ""
```

3 Dudas

En caso de tener dudas específicas de su ejercicio que es **COMPLETAMENTE OPCIONAL**, podrán enviarlas por correo electrónico con el asunto **[Geometría Computacional]** incluyendo los corchetes a la dirección `ailynrp12@ciencias.unam.mx` donde se atenderán a la brevedad posible.