Problema 1

In [ ]:
```python
import random

def juego_individual(prob_beto):
    return random.random() < prob_beto # random entre 0 y 1 (gana o pierde)

def simular_torneo(prob_beto):

    victorias_beto = 0
    victorias_enrique = 0
    juegos_seguidos_beto = 0
    juegos_seguidos_enrique = 0
    juegos_totales = 0

    while True:
        juegos_totales += 1
        if juego_individual(prob_beto):
            victorias_beto += 1
            juegos_seguidos_beto += 1
            juegos_seguidos_enrique = 0
        else:
            victorias_enrique += 1
            juegos_seguidos_enrique += 1
            juegos_seguidos_beto = 0

        if victorias_beto == 3 or juegos_seguidos_beto == 2:
            return True, juegos_totales
        if victorias_enrique == 3 or juegos_seguidos_enrique == 2:
            return False, juegos_totales

def simular_multiples_torneos(num_simulaciones, prob_beto):
    victorias_beto = 0
    victorias_enrique = 0
    total_juegos = 0

    for _ in range(num_simulaciones):
        beto_gano, juegos = simular_torneo(prob_beto)
        if beto_gano:
            victorias_beto += 1
        else:
            victorias_enrique += 1
        total_juegos += juegos

    prob_victoria_beto = victorias_beto / num_simulaciones
    prob_victoria_enrique = victorias_enrique / num_simulaciones
    promedio_juegos = total_juegos / num_simulaciones

    return prob_victoria_beto, prob_victoria_enrique, promedio_juegos
```

In [ ]:
```python
prob_beto = 1/4  # Probabilidad de que Beto gane 1 juego de cada 4
num_simulaciones = 1000000
```

```
prob_victoria_beto, prob_victoria_enrique, promedio_juegos = simular_multiples_torn

print(f"1. Probabilidad de que Beto gane el torneo: {prob_victoria_beto:.4f}")
print(f"   Probabilidad de que Enrique gane el torneo: {prob_victoria_enrique:.4f}"
print(f"2. Número esperado de juegos en el torneo: {promedio_juegos:.4f}")
```

```
1. Probabilidad de que Beto gane el torneo: 0.1381
   Probabilidad de que Enrique gane el torneo: 0.8619
2. Número esperado de juegos en el torneo: 2.6314
```

Problema 2

In [ ]:
```
def simular_cruce(prob_detenido):
    return random.random() < prob_detenido

def simular_camino(prob_detenido, total_cruces):
    cruces_min_tarde = total_cruces / 2
    cruces_detenidos = sum(simular_cruce(prob_detenido) for _ in range(total_cruces
    return cruces_detenidos >= cruces_min_tarde

def simular_viaje(num_simulaciones, prob_detenido, total_cruces):
    tardanzas = sum(simular_camino(prob_detenido, total_cruces) for _ in range(num_
    prob_tarde = tardanzas / num_simulaciones
    return prob_tarde
```

In [ ]:
```
prob_detenido = 0.1
num_simulaciones = 1000000

prob_tarde_ruta_1 = simular_viaje(num_simulaciones, prob_detenido, 4)
prob_tarde_ruta_2 = simular_viaje(num_simulaciones, prob_detenido, 2)

print(f"Probabilidad de llegar tarde por la Ruta 1 (4 cruces): {prob_tarde_ruta_1:.
print(f"Probabilidad de llegar tarde por la Ruta 2 (2 cruces): {prob_tarde_ruta_2:.

if prob_tarde_ruta_1 < prob_tarde_ruta_2:
    print("El profesor Deviation debería tomar la Ruta 1 para minimizar la probabil
else:
    print("El profesor Deviation debería tomar la Ruta 2 para minimizar la probabil
```

```
Probabilidad de llegar tarde por la Ruta 1 (4 cruces): 0.0525
Probabilidad de llegar tarde por la Ruta 2 (2 cruces): 0.1904
El profesor Deviation debería tomar la Ruta 1 para minimizar la probabilidad de lleg
ar tarde.
```

Problema 3

In [ ]:
```
import numpy as np

X = np.array([1, 2, 3, 4, 5, 6])
P_X = np.array([1/15, 2/15, 3/15, 4/15, 3/15, 2/15])
costo_compra = 2
precio_venta = 4

def calcular_ingreso_esperado(n_ejemplares):
    ingresos = np.minimum(X, n_ejemplares) * precio_venta - n_ejemplares * costo_co
```

```python
    return np.dot(ingresos, P_X)

# Calcular ingresos esperados para 3, 4, 5 y 6 ejemplares
ingresos_esperados = [calcular_ingreso_esperado(n) for n in range(3, 7)]

# Calcular el valor esperado de X
E_X = np.dot(X, P_X)

print("Resultados:")
for n, ingreso in zip(range(3, 7), ingresos_esperados):
    print(f"Ingreso esperado para {n} ejemplares: ${ingreso:.2f}")

print(f"\nValor esperado de X (demanda promedio): {E_X:.2f}")

mejor_opcion = 3 + np.argmax(ingresos_esperados)
print(f"\nA) La mejor opción es ordenar {mejor_opcion} ejemplares.")

print("\nComparación entre 3 y 4 ejemplares:")
print(f"Diferencia: ${ingresos_esperados[1] - ingresos_esperados[0]:.2f}")

print("\nComparación entre 5 y 6 ejemplares:")
print(f"Diferencia: ${ingresos_esperados[3] - ingresos_esperados[2]:.2f}")

print("\nB) El pequeño mercado tiene la disyuntiva de comprar 3 o 4 revistas y no 5
print("-El ingreso esperado es mayor para 3 o 4 ejemplares.")
print(f"-La demanda promedio ({E_X:.2f}) está más cerca de 3 o 4 que de 5 o 6.")
print("-Comprar más revistas de las que se espera vender aumenta el riesgo de pérdi
```

```
Resultados:
Ingreso esperado para 3 ejemplares: $4.93
Ingreso esperado para 4 ejemplares: $5.33
Ingreso esperado para 5 ejemplares: $4.67
Ingreso esperado para 6 ejemplares: $3.20

Valor esperado de X (demanda promedio): 3.80

A) La mejor opción es ordenar 4 ejemplares.

Comparación entre 3 y 4 ejemplares:
Diferencia: $0.40

Comparación entre 5 y 6 ejemplares:
Diferencia: $-1.47

B) El pequeño mercado tiene la disyuntiva de comprar 3 o 4 revistas y no 5 o 6 porqu
e:
-El ingreso esperado es mayor para 3 o 4 ejemplares.
-La demanda promedio (3.80) está más cerca de 3 o 4 que de 5 o 6.
-Comprar más revistas de las que se espera vender aumenta el riesgo de pérdidas por
inventario no vendido.
```