Guía de Ejercicios Prácticos

A continuación se plantean una serie de fragmentos de código. Habrá que realizar un análisis de algoritmo para determinar su coste computacional y obtener su orden.

1. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
```

2. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
    if(num % 2 == 0):
        print('es divisible por 2')
    else:
        print('es divisible por 2')
        num = num + 1
        aux = num * num * num
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
    if(producto % 3 == 0):
        print('es divisible por 3')
```

3. Fragmento de código de ciclo:

```
for i in range(0, n):
    print('incio ciclo j')
    for j in range(0, m):
        print('incio ciclo k')
        for k in range(0, 3):
            print(lista[k])
```

4. Fragmento de código de ciclo:

```
for i in range(0, n):
    if(aux is not None):
        print('incio ciclo j')
        for j in range(0, m):
            print(num + (j * valor))
    else:
        print('incio ciclo k')
        for k in range(0, 1000):
            print(aux[k] + num)
```

5. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
    print('tabla de multiplicar')
    for i in range(1, 16):
        print(i, num * i)
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
    if(producto % 3 == 0):
        print('es divisible por 3')
```

6. Fragmento de código:

```
numero = int(input('ingrese un número'))
while (numero ≠ 0) and (len(lista) < 10000):
    lista.append(numero)
    numero = int(input('ingrese número'))

for i in range(0, len(lista)):
    print(lista[i])
```

7. Búsqueda binaria:

```
while (p ⩽ u) and (pos == -1):
    med = (p + u) // 2
    if(lista[med] == x):
        pos = med
    else:
        if(x > lista[med]):
            p = med + 1
        else:
            p = med - 1
```

8. Fragmento de código:

```
while (aux is not None):
    print(aux.info)
    aux = aux.sig
```

9. Fragmento de código:

```
for i in range(0, n):
    ac = 0
    for j in range(0, m):
        ac = ac + lista[i, j]
    aux[i] = ac

i = 0
while (i < len(aux)):
    print(aux[i])
    i += 1
```

10. Fragmento de código:

```
for i in range(0, n-1):
    min = lista[inicio, n-1]
    pos_min = n-1
    if(min == 0):
        min = lista[inicio, n-2]
        pos_min = n-2
    for j in range(0, n-1):
        if(min > lista[inicio, j] and lista[inicio, j] != 0
           and j not in camino):
            min = lista[inicio, j]
            pos_min = j
    camino.append(pos_min)
    inicio = pos_min
```

11. Multiplicación de dos matrices M1[n, m] y M2[m, o].
12. Función factorial recursiva.
13. Función fibonacci recursiva.
14. Función fibonacci iterativa.
15. Búsqueda binaria recursiva.