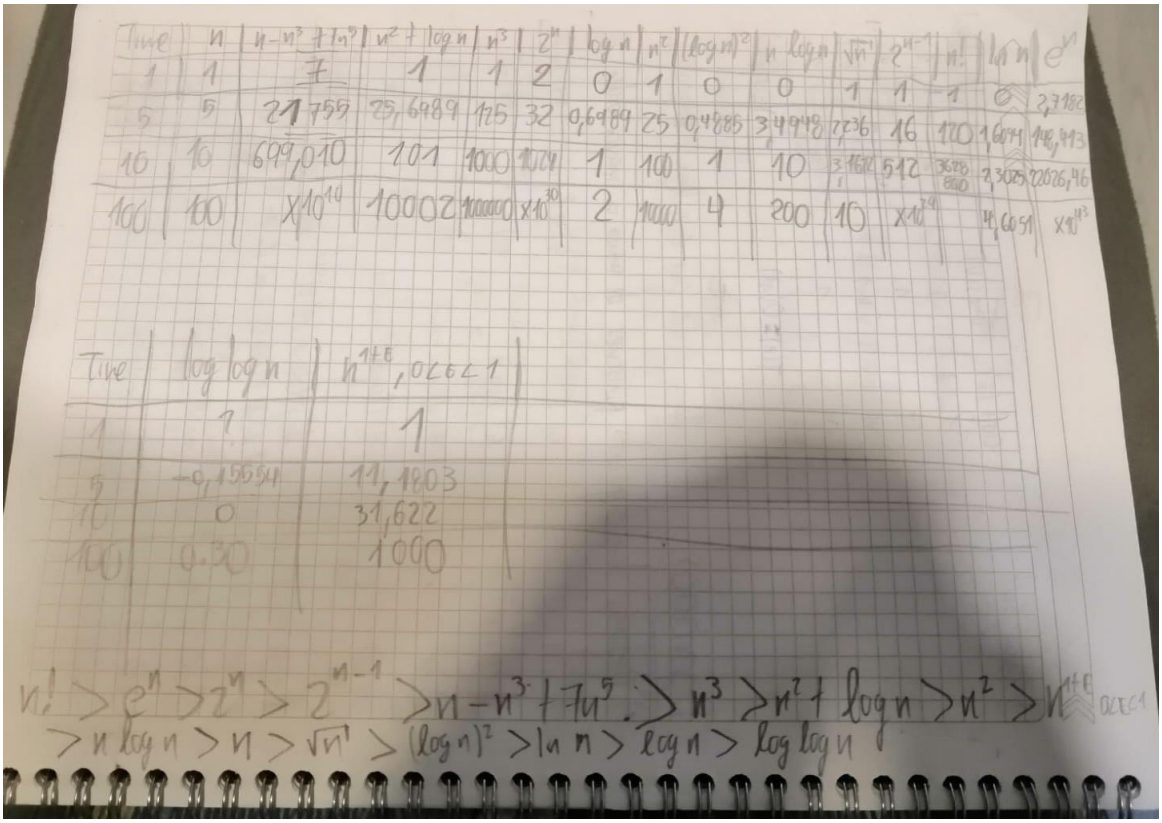
ANDERSSON DAVID SÁNCHEZ MÉNDEZ **CONTROL DE LECTURA 1**

Ejercicios

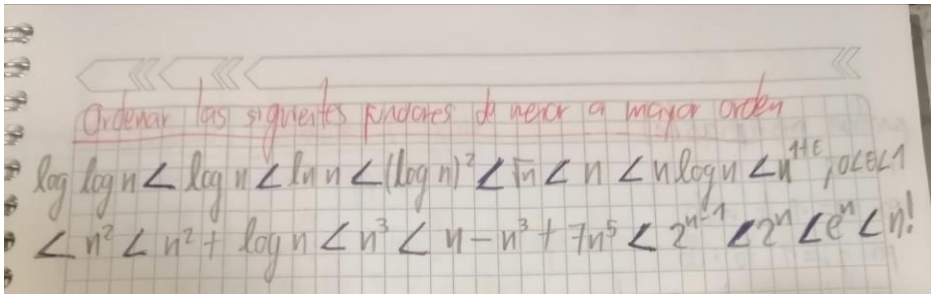
1. Ordenar las siguientes funciones de menor a mayor orden:

1. n	5. 2^n	9. $n \log n$	13. $\ln n$
2. $n - n^3 + 7n^5$	6. $\log n$	10. $\sqrt{n}$	14. e^n
3. $n^2 + \log n$	7. n^2	11. 2^{n-1}	15. $\log \log n$
4. n^3	8. $(\log n)^2$	12. $n!$	16. $n^{1+\varepsilon}, 0 < \varepsilon < 1$



MÉTODO DE TABLA DE VALORES

Así:



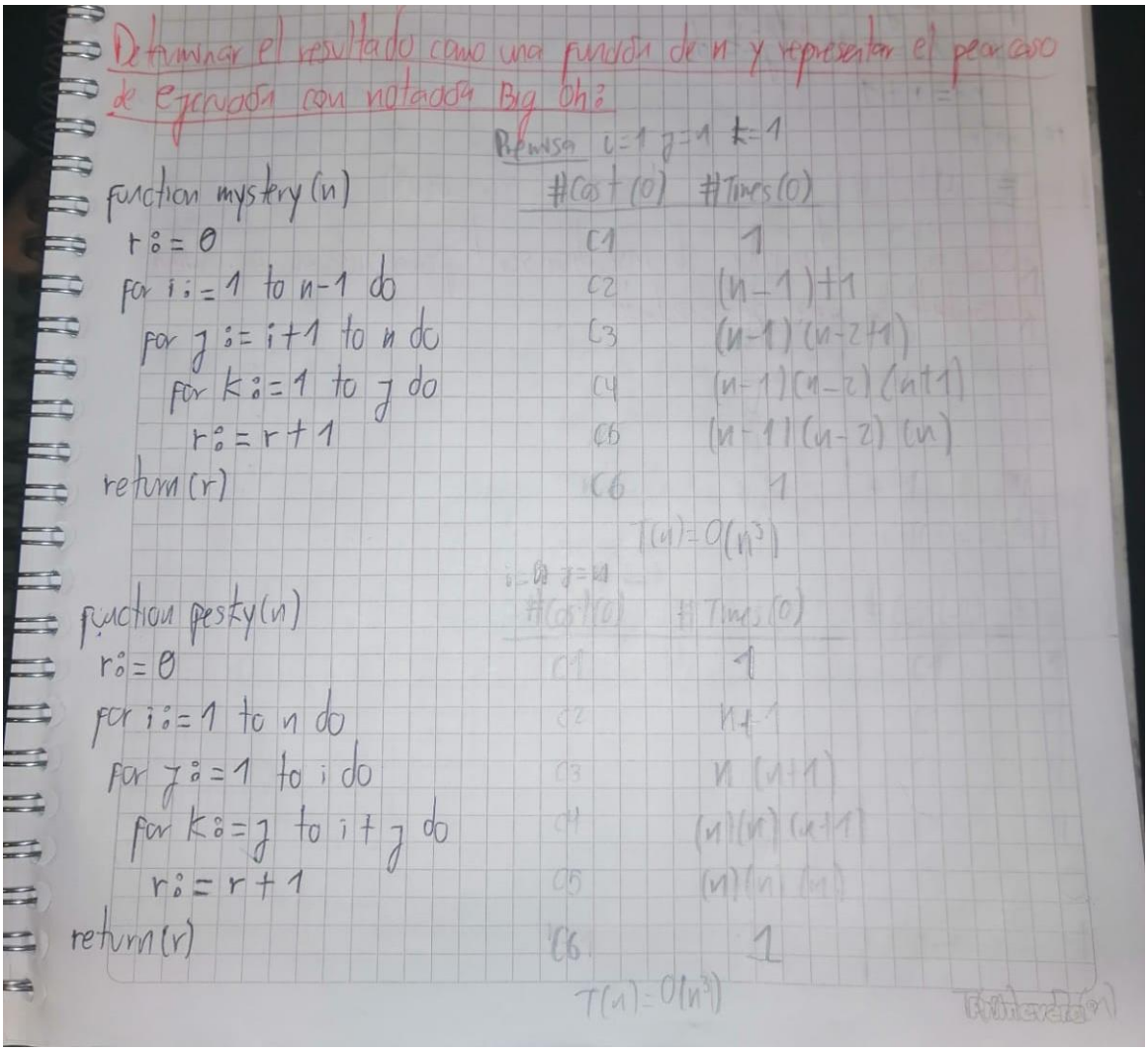
2. Para las siguientes funciones, determinar el resultado como una función de *n* y representar el peor caso de ejecución con notación Big Oh:

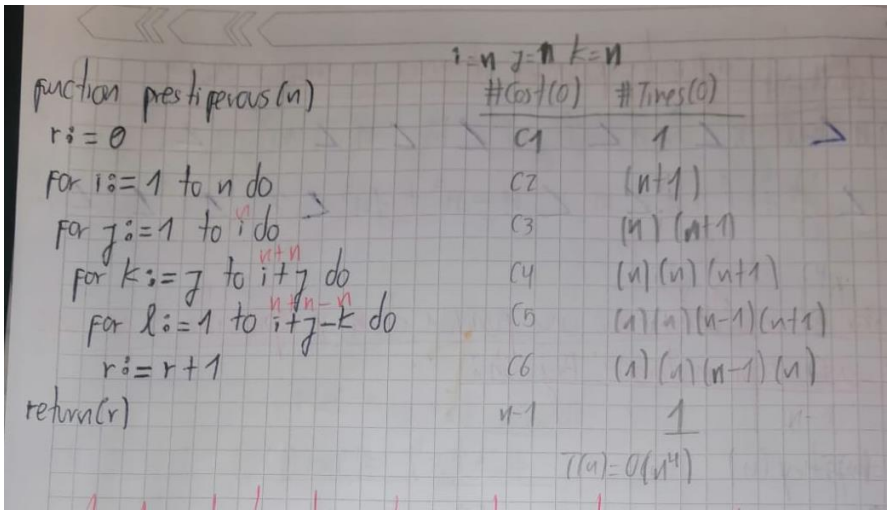
```
function mystery(n)
  r := 0
  for i := 1 to n - 1 do
    for j := i + 1 to n do
      for k := 1 to j do
        r := r + 1
  return(r)
```

```
function pesky(n)
  r := 0
  for i := 1 to n do
    for j := 1 to i do
      for k := j to i + j do
        r := r + 1
  return(r)
```

```
function prestiferous(n)
  r := 0
  for i := 1 to n do
    for j := 1 to i do
      for k := j to i + j do
        for l := 1 to i + j - k do
          r := r + 1
  return(r)
```

★ $\sum_{x=1}^{n} x = \frac{1}{2} n(n+1)$ $\sum_{j=i+1}^{n} j = \sum_{j=1}^{n} j - \sum_{j=1}^{i} j$ $\sum_{x=1}^{n} x^2 = \frac{1}{6} n(n+1)(2n+1)$





3. Implementar el algoritmo de *insertion sort* para ordenar en orden descendente en vez de ascendente.

```python
START=1

def addElement(element, seq):
    element_index = 0
    while element_index < len(seq) and not (seq[element_index] > element):
        element_index = element_index + 1
    return seq[:element_index] + [element] + seq[element_index:]

def insertionSort(seq):
    result = seq[:START]
    for e in range(START, len(seq)):
        result = addElement(seq[e], result)
    return result[::-1]

input_list = [5, 8, -1, 4, 6]

print(insertionSort(input_list))
```

```
Run  BUSCAMINAS  CLASS MATERIAL 2
C:\Users\ANDERSSON\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users
[8, 6, 5, 4, -1]

Process finished with exit code 0
```