# ALGORÍSMICA I – CURS 2009/2010

**PROVA DESEMBRE**

Creeu un únic mòdul, **prova.py**, amb les dues funcions següents (la primera val 4/10 i la segona 6/10):

**Enunciat**:

Escriu una funció (**exponent**) que usant tècniques de dividir i vèncer, calculi el valor $a^n$ per qualsevol $a > 0$ , sent $n$ un enter positiu.

**Exemple** del comportament d'aquesta funció:

```
>>> exponent(3,5) # a=3 , n=5
243
>>> exponent(3.5,5) # a=3.5 , n=5
525.21875
>>> exponent(45.78,45)
5.3753961414860256e+74
```

**Avaluació**: A l'avaluar aquest exercici tindrem en compte els següents punts:

- Si el programa no usa una estratègia de dividir i vèncer, la nota és 0.

**Indicacions**:

Podeu fer servir el fet que $a^n = a^{\lfloor n/2 \rfloor} a^{\lceil n/2 \rceil}$

**Enunciat**:

Escriu una funció (**repeticions**) que conti quantes vegades apareix un nombre enter *x* en una llista ordenada de nombres enters i que tingui complexitat *O*(log*n* + *m*), on *m* és el nombre de vegades que es repeteix el nombre.

**Exemple** del comportament d'aquesta funció:

```
>>> A=[1,2,3,4,4,4,5,6,7,8,9,9,9,9,10]
>>> B=[1,1,1,1]
>>> repeticions(B,1)
4
>>> repeticions(A,10)
1
>>> repeticions(A,9)
4
>>> repeticions(A,0)
'El nombre no hi es'
>>> repeticions(A,1)
1
```

**Avaluació**: A l'avaluar aquest exercici tindrem en compte els següents punts:

- Si el programa dona un resultat correcte a partir d'una versió de la cerca binària: 7 punts sobre 10.
- Si el programa és concís: 2 punts sobre 10.
- Ús adequat del llenguatge (`if/while`, etc), bon estil de programació, etc.: 1 punts sobre 10.

**Indicacions**:

Partiu de l'algorisme de la cerca binària i modifiqueu-lo.

# ALGORÍSMICA I – CURS 2009/2010

**PROVA DESEMBRE**

Creeu un únic mòdul, **prova.py**, amb les dues funcions següents (la primera val 4/10 i la segona 6/10):

**Enunciat**:

Escriu una funció (**reverse**) que usant tècniques de dividir i vèncer, inverteixi l'ordre dels caràcters d'un string.

**Exemple** del comportament d'aquesta funció:

```
>>> reverse("hola, com estas?")
"?satse moc ,aloh"
```

**Avaluació**: A l'avaluar aquest exercici tindrem en compte els següents punts:

- Si el programa no usa una estratègia de dividir i vèncer, la nota és 0.

**Enunciat**:

Escriu una funció (**centre**) que, amb complexitat *O*(log*n*), donat un vector *A* ordenat i un enter *b*, imprimeixi la quina posició *i* del vector tal que es compleix que l'element a la seva posició i tots els elements a la dreta són iguals o menors que *b;* i els de la dreta més grans:

$$\forall j, si\ j \leq i, A[j] \leq b$$
$$\forall j, si\ j > i, A[j] > b$$

**Exemple** del comportament d'aquesta funció:

```
>>> A=[1,2,4,4,4,6,7,9,9,9,9,10]
>>> centre(A,7)
6
>>> centre(A,5)
4
>>> centre(A,1)
0
>>> centre(A,10)
11
>>>
```

**Avaluació**: A l'avaluar aquest exercici tindrem en compte els següents punts:

- Si el programa dona un resultat correcte a partir de la modificació de la cerca binària: 7 punts sobre 10.
- Ús adequat del llenguatge (`if`/`while`, etc): 2 punts sobre 10.
- Bon estil de programació (fer una interfase d'usuari adequada, comentaris, etc.): 1 punt sobre 10.

**Indicacions**:

Partiu de l'algorisme de la cerca binària recursiu i modifiqueu-lo lleugerament.

# Solucions

Exercici 1.1:

```
def exponent(a,n):
    import math
    if n==1: return a
    else: return exponent(a,math.floor(n/2.0))*exponent(a,math.ceil(n/2.0))
```

Exercici 2.1:

```
def binsearch(A, value):
    low = 0
    high = len(A)-1
    while low <= high:
        mid = (low + high) / 2;
        if A[mid] > value:
            high = mid - 1;
        elif A[mid] < value:
            low = mid + 1;
        else:
            return mid;
    return -1

def repeticions(A, value):
    cont=0
    pos=binsearch(A,value)
    pos1,pos2=pos, pos
    if pos==-1: return "El nombre no hi es"
    else:
        while A[pos1]==value and pos1>0:
            pos1=pos1-1
            cont=cont+1
        while A[pos2]==value and pos2<len(A)-1:
            pos2=pos2+1
            cont=cont+1
    if A[0]==value: pos1=pos1-1
    if A[len(A)-1]==value: pos2=pos2+1
    print pos1+1, pos2-1, (pos2-1)-(pos1+1)+1
```

Exercici 1.2:

```
def reverse(a):
    if a=="": return a
    return reverse(a[1:])+a[0]
```

Exercici 2.2:

```
def search(x, nums, low, high):
    mid = (low + high) / 2;
    if low>high: return low-1
    elif x < nums[mid]:
           return search(x,nums,low,mid-1)
    else: return search (x,nums,mid+1,high)

def centre(nums,x):
    return search(x,nums,0,len(nums)-1)
```