ALGORÍSMICA I – CURS 2009/2010

PROVA NOVEMBRE

Creeu un únic mòdul, **prova.py**, amb les dues funcions següents (cada una val 5/10 punts):

1. Escriu una funció (**negatius**) amb les següents indicacions:

Enunciat:

Donada una llista de nombres enters (possiblement desordenats) la funció ha de retornar la mateixa llista amb els nombres negatius al capdavant i els positius al darrera (sense importar l'ordre entre ells).

Exemple del comportament d'aquesta funció:

```
>>> a=[1,-2,3,-4,-3,5,6]
>>> negatius(a)
>>> a
[-6,-2,-3,-4,5,3,1]
```

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

- Si el programa dona un resultat correcte: 4 punts sobre 10.
- Si el programa s'executa sobre una mateixa llista (és a dir, no crea cap llista auxiliar a l'executar-se): 2 punts sobre 10.
- Si la complexitat de l'algorisme que proposeu és $O(n)$: 2 punts sobre 10.
- Ús adequat del llenguatge (`if`/`while`, etc): 1 punt sobre 10.
- Bon estil de programació (fer una interfase d'usuari adequada, comentaris, etc.): 1 punt sobre 10.

Indicacions:

Aquest algorisme es pot resoldre de la forma demanada amb una estratègia semblant (tot i que una mica més simple) a la **partició** del *quicksort*.

2. Escriu una funció (**asterisc**) amb les següents indicacions:

Enunciat:

Modifica l'algorisme de **Horspool** de manera que: (1) consideri el caràcter "*" com a comodí (que vol dir que aquest caràcter pot correspondre a qualsevol caràcter) i (b) ens retorni totes les posicions on hi ha el primer *string*.

Exemple del comportament d'aquesta funció:

```
>>> a="c*sa"
>>> b="la casa del costat de la cosa"
>>> asterisc(a,b)
3 14
```

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

- Si el programa dona un resultat correcte: 6 punts sobre 10.
- Si el programa fa els canvis mínims a la versió normal de l'algorisme de Hospool: 2 punts sobre 10.
- Ús adequat del llenguatge (`if`/`while`, etc), bon estil de programació (fer una interfase d'usuari adequada, comentaris, etc.): 2 punts sobre 10.

ALGORÍSMICA I – CURS 2009/2010

PROVA NOVEMBRE

Creeu un únic mòdul, **prova.py**, amb les següents funcions (cada una val 5/10 punts):

1. Escriu una funció (**maxim**) amb les següents indicacions:

Enunciat:

La funció s'ha d'escriure amb l'estratègia de "dividir i vèncer" i ha de retornar el nombre més gran d'una llista d'enters.

Exemple del comportament d'aquesta funció:

```
>>> a=[1,-2,3,-4,-3,5,66,-3,9,-67]
>>> maxim(a)
El maxim es 66.
```

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

- Si el programa dona un resultat correcte: 4 punts sobre 10.
- Si el programa s'executa sobre una mateixa llista (és a dir, no crea cap llista auxiliar a l'executar-se): 2 punts sobre 10.
- Ús adequat del llenguatge (`if/while`, etc): 3 punts sobre 10.
- Bon estil de programació (fer una interfase d'usuari adequada, comentaris, etc.): 1 punt sobre 10.

2. Escriu una funció (**cadena**) amb les següents indicacions:

Enunciat:

Donats dos *strings*, ens digui si el primer *string* és igual a una possible reordenació dels caràcters centrals del segon *substring*. Per caràcters centrals entenem aquells que no són ni el primer ni l'últim: els caràcters centrals de "`algorismes`" són "`lgorisme`".

Exemple del comportament d'aquesta funció:

```
>>>a="comparacio"
>>>b="cciomrapao"
>>>cadena(a,b)
True
>>>a="12345"
>>>b="54321"
>>>cadena(a,b)
False
```

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

- Si el programa dona un resultat correcte: 6 punts sobre 10.
- Si el programa genera el mínim nombre de comparacions: 2 punts.
- Ús adequat del llenguatge (`if/while`, etc): 1 punt sobre 10.
- Bon estil de programació (fer una interfase d'usuari adequada, comentaris, etc.): 1 punt sobre 10.

Indicacions:

Aquí teniu una funció recursiva per generar permutacions d'un *string*:

```
def allperm(str):
  if len(str) <= 1:
     return str
  else:
     biglist = []
     for perm in allperm(str[1:]):
        for i in range(len(str)):
           biglist.append(perm[:i] + str[0:1] + perm[i:])
  return biglist
```

Solucions

Exercici 1:

```
def negatius(a):
    i=0
    j=len(a)-1
    while i <= j:
           if a[i]<0: i += 1
           else:
                  a[i],a[j]= a[j],a[i]
                  j -= 1
```

Exercici 2:

```
def asterisc(pattern, text):
    m = len(pattern)
    n = len(text)
    if m > n: return -1
    skip = []
    for k in range(256): skip.append(m)
    for k in range(m - 1):
        if pattern[k] != "*": skip[ord(pattern[k])] = m - k - 1
    skip = tuple(skip)
    k = m - 1
    while k < n:
        j = m - 1; i = k
        while j >= 0 and (text[i] == pattern[j] or pattern[j] == "*"):
            j -= 1; i -= 1
        if j == -1: print i + 1,
        k += skip[ord(text[k])]

a="c*sa"
b="la casa del costat de la cosa"
asterisc(a,b)
```

Exercici 3:

```
def maxim(a):
    if len(a) < 2:
        return a[len(a)-1]
    else:
        centre = len(a) / 2
        temp1 = maxim(a[centre:])
        temp2 = maxim(a[:centre])
        if temp1>=temp2:
            return temp1
        else: return temp2

a=[3,5,7,8,33,55,1,3,5,7]
print maxim(a)
```

Exercici 4:

```
def allperm(str):
  if len(str) <= 1:
     return str
  else:
     biglist = []
     for perm in allperm(str[1:]):
        for i in range(len(str)):
           biglist.append(perm[:i] + str[0:1] + perm[i:])
  return biglist


def cadena(a,b):
  res="False"
  for x in allperm(a[1:len(a)-1]):
    if a[0]+x+a[len(a)-1]==b: res="True"
  print res


a="comparacio"
b="cciomrapao"
cadena(a,b)
```