Carthago delenda est

Des de Roma volen enviar una sèrie de missatges a Tarraco sense que Cartago els pugui interceptar, i necessiten que es pugui encriptar de forma ràpida.

Ens demanen de fer un codificador de text per transmetre missatges xifrats, on tant el missatge original com l'encriptat tenen com a vocabulari les 23 lletres de l'alfabet llatí, tenint com a clau un enter, i que no costi gaire de computar.

Exemple

Per exemple, donat el text: "quo usque tandem abutere, Catilina, patientia nostra", la seva encriptació seguint el mètode de Cèsar és: "tyr yvtyh xdqghp deyxhuh, Fdxnonqd, sdxnhqxnd qrvxud".

Requeriments

Es demana realitzar:

- Un programa que llegeixi les dades d'un fitxer de text i retorni el text encriptat, i un altre que llegeixi l'encriptat i retorni el text en clar. Tant l'encriptació com la desencriptació han de tenir un cost de $O(n)$. La codificació haurà de superar el test del `makefile`, que comprova si:
  - tant el text del text com de l'encriptat són de l'abecedari llatí (considerarem aquest com "*abcdefghilmnopqrstuvxyz*"), heu de mantenir si són majúscules o minúscules i els signes de puntuació,
  - es pot detectar la clau amb substitució simple o transposició simple.
- Un informe que contingui, sobre els algorismes principals: pseudo-codi en disseny recursiu i iteratiu, cost teòric i experimental.

Arguments i parametres

L'execuci'o de l'aplicació haurà de seguir la següent sintaxi:

```
$./encrypt clau [fitxer en clar] [fitxer codificat]
$./decrypt clau [fitxer codificat] [fitxer en clar]
```

El arguments obligatoris són els següets:

clau : un enter.

Els arguments opcionals del programa són els següents:

fitxer en clar : Fitxer sense codificar.

fitxer codificat : Fitxer codificat segons el vostre mètode.

Si no hi ha arguments opcionals, l'aplicació llegeix de l'entrada estàndard i escriu a la sortida estàndard.

Avaluació

En l'informe de la pràctica, els algorismes principals, les taules i les gràfiques han d'estar comentats.

Els programes han de compilar sense errors ni warnings i passar els tests amb `make test`, altrament no s'avaluarà.

La baremació de la pràctica (sobre 10) serà la següent:

Costos 3 punts, anàlisi de costos teòrics (recursiu i iteratiu) i empírics dels algorismes;

Disseny 3 punts, recursiu i iteratiu (es tindrà en compte el cost de l'algorisme);

Implementació 4 punts

- 1 punt, iteratiu en C
- 1 punt, recursiu en python
- 1 punt, bones pràctiques de programació
- 1 punt, utilització dels recursos del llenguatge de programació.
- 1 punt extra, recursiu en haskell.

Només per als *alumnes de pla antic*, hi ha un treball extra sobre la pràctica:

Especificació 1 punt, especificació formal de l'encriptador;

Especificació 1 punt, especificació formal del desencriptador;

Enviament

L'assignació és per parelles, i representa un 20% de la nota final. Presenteu la pràctica al Campus Virtual de la UdL amb dos fitxers: un pdf per a l'informe, i un arxiu comprimit, `tgz` o `zip` (no s'admeten altres formats de compressió), per al codi font.

Els programes que s'hagin de compilar ho han de fer amb:

```
$ make
```

Els programes es testejen fent:

```
$ make test
```

La pràctica podria ser verificada individualment el primer dia de laboratori després de l'entrega.