Esame 20230120

Esercizio 1

(1) **Esercizio 1 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio1.cc` un programma che prenda come argomento del main:

- Il nome di un file di testo in input, contenente una sequenza di parole separate da spazi, di soli caratteri *0...9,a...z* (10 cifre e 26 lettere minuscole).
- Il nome di un file di testo da usare per l'output.

Il programma chiede all'utente un numero di al massimo 7 cifre da usare come **chiave di criptazione** del file letto. Richiedere continuamente all'utente la chiave se il numero immesso ha più di 7 cifre.

Il programma quindi legge il contenuto del file di input parola per parola **troncando al quarto carattere**, e.g. *abcde* diventa *abcd*.

Decodifica la parola nella sua rappresentazione decimale, effettuando una trasformazione dalla base 36 alla base 10. La **base 36** usa le cifre *0...9* e (ventisei) caratteri minuscoli *a...z*, che assumono valori compresi tra 0 e 35, ad esempio: $0_{36} \rightarrow 0$, $9_{36} \rightarrow 9$, $a_{36} \rightarrow 10$, $z_{36} \rightarrow 35$. Data una parola/numero in base 36 sulla destra abbiamo la cifra/carattere meno significativo e poi man mano potenze di 36 crescenti. Ad esempio, $abc_{36} \rightarrow 10*(36^2)+11*(36^1)+12*(36^0) = 13368$, altri esempi sono $1z_{36} \rightarrow 71$, $20_{36} \rightarrow 72$, $21_{36} \rightarrow 73$.

La chiave intera precedentemente acquisita viene quindi sommata ad ogni parola decodificata per criptarla.

Una volta criptata, la parola va **ricodificata** nella sua rappresentazione originale in base 36, quindi cifre 0...9 e lettere minuscole a...z, e stampata nel file di output.

Ad esempio, se la chiave inserita vale 1, a_{36} diventa b_{36}, mentre $1z_{36}$ (=71) diventa 20_{36} (=72).

Per fare la ricodifica, effettuare una trasformazione dalla base 10 alla base 36 per codificare le parole, ad esempio $13368 \rightarrow abc_{36}$. Qui un possibile procedimento: (1) Il primo carattere/cifra a destra si ottiene facendo il modulo 36 del numero, e.g. $13368\%36 = 12$. (2) Quindi si rappresenta il valore ottenuto con il relativo carattere/cifra, e.g. $12 \rightarrow c_{36}$. (3) Poi, sottratto il modulo si divide per 36, e.g. $(13368 - 12)/36 = 371$. (4) Si ripete da (1) fino ad avere resto 0. (5) Al termine riordinare le cifre dalla più significativa a sinistra alla meno significativa a destra. Ogni cifra puó quindi essere usata per trasformarla nel carattere corrispondente per la base 36 (secondo la codifica specificata con 0,...,9,a,...,z).

Supponiamo che il primo file `input.txt` contenga

```
sole mare luna mondo
```

Eseguando il programma `./esercizio1.out input.txt output.txt` verrá chiesta all'utente la chiave per criptare il testo. Inserendo **36** il programma dovrá produrre un file chiamato `output.txt` che conterrá i seguenti valori:

```
some mase luoa mood
```

Note:

- Si assuma che il file di input includa solo cifre 0...9 e caratteri a...z. Si noti che 36 é dato dal conteggio delle cifre 0...9 e dei caratteri *a...z*, dove quindi 'z' - 'a' = 25.
- È ammesso l'uso della libreria cstring, e.g. `strlen`.
- È consentito l'utilizzo della libreria cmath, e.g. la funzione `pow`.
- Non è consentito l'utilizzo di altre funzioni di libreria "particolari" diverse da quelle specificate sopra o da quelle standard necessarie a risolvere l'esercizio.
- Si noti che le parole in input devo essere troncate per evitare problemi di overflow, lo stesso vale per la chiave numerica chiesta all'utente che deve soddifare la lunghezza specificata nel testo.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

esercizio1.cc

Information for graders:

(2) **Esercizio 1 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio1.cc` un programma che prenda come argomento del main:

- Il nome di un file di testo in input, contenente una sequenza di parole di sole cifre 0 o caratteri minuscoli a...z separate da spazi.
- Il nome di un file di testo da usare per l'output.

Il programma chiede all'utente un numero di al massimo 7 cifre da usare come **chiave di criptazione** del file letto. Richiedere continuamente all'utente la chiave se il numero immesso ha più di 7 cifre.

Il programma quindi legge il contenuto del file di input parola per parola **troncando al quarto carattere**, e.g. *abcde* diventa *abcd*.

Decodifica la parola nella sua rappresentazione decimale, effettuando una trasformazione dalla base 27 alla base 10. La **base 27** usa la cifre *0* e (ventisei) caratteri minuscoli *a...z*, che assumono valori compresi tra 0 e 26, ad esempio: $0_{27} \rightarrow 0$, $a_{27} \rightarrow 1$, $z_{27} \rightarrow 26$. Data una parola/numero in base 27 sulla destra abbiamo la cifra/carattere meno significativo e poi man mano potenze di 27 crescenti. Ad esempio, $abc_{27} \rightarrow 1 * (27^2) + 2 * (27^1) + 3 * (27^0) = 786$, altri esempi sono $a0_{27} \rightarrow 27$, $aa_{27} \rightarrow 28$, $az_{27} \rightarrow 53$, $b0_{27} \rightarrow 54$, $ba_{27} \rightarrow 55$.

La chiave intera precedentemente acquisita viene quindi sommata ad ogni parola decodificata per criptarla.

Una volta criptata, la parola va **ricodificata** nella sua rappresentazione originale in base 27, quindi la cifra 0 e le lettere minuscole a...z, e stampata nel file di output.

Ad esempio, se la chiave inserita vale 1, a_{27} diventa b_{27}, mentre az_{27} (=53) diventa $b0_{27}$ (=54).

Per fare la ricodifica, effettuare una trasformazione dalla base 10 alla base 27 per codificare le parole, ad esempio $786 \rightarrow abc_{27}$. Qui un possibile procedimento: (1) Il primo carattere/cifra a destra si ottiene facendo il modulo 27 del numero, e.g. $786\%27 = 3$. (2) Quindi si rappresenta il valore ottenuto con il relativo carattere/cifra, e.g. $3 \rightarrow c_{27}$. (3) Poi, sottratto il modulo si divide per 27, e.g. $(786 - 3)/27 = 29$. (4) Si ripete da (1) fino ad avere resto 0. (5) Al termine riordinare le cifre dalla più significativa a sinistra alla meno significativa a destra. Ogni cifra puó quindi essere usata per trasformarla nel carattere corrispondente per la base 27 (secondo la codifica specificata con 0,a,...,z).

Supponiamo che il primo file `input.txt` contenga

```
sole mare luna mondo
```

Eseguando il programma `./esercizio1.out input.txt output.txt` verrá chiesta all'utente la chiave per criptare il testo. Inserendo **27** il programma dovrá produrre un file chiamato `output.txt` che conterrá i seguenti valori:

```
some mase luoa mood
```

Note:

- Si assuma che il file di input includa solo 0 e caratteri a...z. Si noti che 27 é dato dal conteggio dei caratteri 0,a,...,z, dove 'z' - 'a' = 25.

- È ammesso l'uso della libreria cstring, e.g. `strlen`.
- È consentito l'utilizzo della libreria cmath, e.g. la funzione `pow`.
- Non è consentito l'utilizzo di altre funzioni di libreria "particolari" diverse da quelle specificate sopra o da quelle standard necessarie a risolvere l'esercizio.
- Si noti che le parole in input devo essere troncate per evitare problemi di overflow, lo stesso vale per la chiave numerica chiesta all'utente che deve soddifare la lunghezza specificata nel testo.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

esercizio1.cc

Information for graders:

(3) **Esercizio 1 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio1.cc` un programma che prenda come argomento del main:

- Il nome di un file di testo in input, contenente una sequenza di parole separate da spazi, di soli caratteri *0...9,a...z* (10 cifre e 26 lettere minuscole).
- Il nome di un file di testo da usare per l'output.

Il programma chiede all'utente un numero di al massimo 5 cifre da usare come **chiave di criptazione** del file letto. Richiedere continuamente all'utente la chiave se il numero immesso ha più di 5 cifre.

Il programma quindi legge il contenuto del file di input parola per parola **troncando al terzo carattere**, e.g. *abcde* diventa *abc*.

Decodifica la parola nella sua rappresentazione decimale, effettuando una trasformazione dalla base 36 alla base 10. La **base 36** usa le cifre *0...9* e (ventisei) caratteri minuscoli *a...z*, che assumono valori compresi tra 0 e 35, ad esempio: $0_{36} \rightarrow 0$, $9_{36} \rightarrow 9$, $a_{36} \rightarrow 10$, $z_{36} \rightarrow 35$. Data una parola/numero in base 36 sulla destra abbiamo la cifra/carattere meno significativo e poi man mano potenze di 36 crescenti. Ad esempio, $abc_{36} \rightarrow 10*(36^2)+11*(36^1)+12*(36^0) = 13368$, altri esempi sono $1z_{36} \rightarrow 71$, $20_{36} \rightarrow 72$, $21_{36} \rightarrow 73$.

La chiave intera precedentemente acquisita viene quindi sommata ad ogni parola decodificata per criptarla.

Una volta criptata, la parola va **ricodificata** nella sua rappresentazione originale in base 36, quindi cifre 0...9 e lettere minuscole a...z, e stampata nel file di output.

Ad esempio, se la chiave inserita vale 1, a_{36} diventa b_{36}, mentre $1z_{36}$ (=36) diventa 20_{36} (=37).

Per fare la ricodifica, effettuare una trasformazione dalla base 10 alla base 36 per codificare le parole, ad esempio $13368 \rightarrow abc_{36}$. Qui un possibile procedimento: (1) Il primo carattere/cifra a destra si ottiene facendo il modulo 36 del numero, e.g. $13368\%36 = 12$. (2) Quindi si rappresenta il valore ottenuto con il relativo carattere/cifra, e.g. $12 \rightarrow c_{36}$. (3) Poi, sottratto il modulo si divide per 36, e.g. $(13368 - 12)/36 = 371$. (4) Si ripete da (1) fino ad avere resto 0. (5) Al termine riordinare le cifre dalla più significativa a sinistra alla meno significativa a destra. Ogni cifra puó quindi essere usata per trasformarla nel carattere corrispondente per la base 36 (secondo la codifica specificata con 0,...,9,a,...,z).

Supponiamo che il primo file `input.txt` contenga

```
sole mare luna mondo
```

Eseguando il programma `./esercizio1.out input.txt output.txt` verrá chiesta all'utente la chiave per criptare il testo. Inserendo **36** il programma dovrá produrre un file chiamato `output.txt` che conterrá i seguenti valori:

```
spl mbr lvn mpn
```

Note:

- Si assuma che il file di input includa solo cifre 0...9 e caratteri a...z. Si noti che 36 é dato dal conteggio delle cifre 0...9 e dei caratteri *a...z*, dove quindi 'z' - 'a' = 25.

- È ammesso l'uso della libreria cstring, e.g. `strlen`.
- È consentito l'utilizzo della libreria cmath, e.g. la funzione `pow`.
- Non è consentito l'utilizzo di altre funzioni di libreria "particolari" diverse da quelle specificate sopra o da quelle standard necessarie a risolvere l'esercizio.
- Si noti che le parole in input devo essere troncate per evitare problemi di overflow, lo stesso vale per la chiave numerica chiesta all'utente che deve soddifare la lunghezza specificata nel testo.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

esercizio1.cc

Information for graders:

(4) **Esercizio 1 v4**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio1.cc` un programma che prenda come argomento del main:

- Il nome di un file di testo in input, contenente una sequenza di parole di sole cifre 0 o caratteri minuscoli a...z separate da spazi.
- Il nome di un file di testo da usare per l'output.

Il programma chiede all'utente un numero di al massimo 5 cifre da usare come **chiave di criptazione** del file letto. Richiedere continuamente all'utente la chiave se il numero immesso ha più di 5 cifre.

Il programma quindi legge il contenuto del file di input parola per parola **troncando al terzo carattere**, e.g. *abcde* diventa *abc*.

Decodifica la parola nella sua rappresentazione decimale, effettuando una trasformazione dalla base 27 alla base 10. La **base 27** usa la cifre *0* e (ventisei) caratteri minuscoli *a...z*, che assumono valori compresi tra 0 e 26, ad esempio: $0_{27} \rightarrow 0$, $a_{27} \rightarrow 1$, $z_{27} \rightarrow 26$. Data una parola/numero in base 27 sulla destra abbiamo la cifra/carattere meno significativo e poi man mano potenze di 27 crescenti. Ad esempio, $abc_{27} \rightarrow 1 * (27^2) + 2 * (27^1) + 3 * (27^0) = 786$, altri esempi sono $a0_{27} \rightarrow 27$, $aa_{27} \rightarrow 28$, $az_{27} \rightarrow 53$, $b0_{27} \rightarrow 54$, $ba_{27} \rightarrow 55$.

La chiave intera precedentemente acquisita viene quindi sommata ad ogni parola decodificata per criptarla.

Una volta criptata, la parola va **ricodificata** nella sua rappresentazione originale in base 27, quindi la cifra 0 e le lettere minuscole a...z, e stampata nel file di output.

Ad esempio, se la chiave inserita vale 1, a_{27} diventa b_{27}, mentre az_{27} (=53) diventa $b0_{27}$ (=54).

Per fare la ricodifica, effettuare una trasformazione dalla base 10 alla base 27 per codificare le parole, ad esempio $786 \rightarrow abc_{27}$. Qui un possibile procedimento: (1) Il primo carattere/cifra a destra si ottiene facendo il modulo 27 del numero, e.g. $786\%27 = 3$. (2) Quindi si rappresenta il valore ottenuto con il relativo carattere/cifra, e.g. $3 \rightarrow c_{27}$. (3) Poi, sottratto il modulo si divide per 27, e.g. $(786 - 3)/27 = 29$. (4) Si ripete da (1) fino ad avere resto 0. (5) Al termine riordinare le cifre dalla più significativa a sinistra alla meno significativa a destra. Ogni cifra puó quindi essere usata per trasformarla nel carattere corrispondente per la base 27 (secondo la codifica specificata con 0,a,...,z).

Supponiamo che il primo file `input.txt` contenga

```
sole mare luna mondo
```

Eseguando il programma `./esercizio1.out input.txt output.txt` verrá chiesta all'utente la chiave per criptare il testo. Inserendo **27** il programma dovrá produrre un file chiamato `output.txt` che conterrá i seguenti valori:

```
spl mbr lvn mpn
```

Note:

- Si assuma che il file di input includa solo 0 e caratteri a...z. Si noti che 27 é dato dal conteggio dei caratteri 0,a,...,z, dove 'z' - 'a' = 25.

- È ammesso l'uso della libreria cstring, e.g. `strlen`.
- È consentito l'utilizzo della libreria cmath, e.g. la funzione `pow`.
- Non è consentito l'utilizzo di altre funzioni di libreria "particolari" diverse da quelle specificate sopra o da quelle standard necessarie a risolvere l'esercizio.
- Si noti che le parole in input devo essere troncate per evitare problemi di overflow, lo stesso vale per la chiave numerica chiesta all'utente che deve soddifare la lunghezza specificata nel testo.
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

esercizio1.cc

Information for graders:

Total of marks: 40