Programmazione Modulo 2:
Ruzzle

L'obiettivo di questa esercitazione è di progettare ed implementare un programma per giocare in maniera automatica a Ruzzle. Nella realizzazione della vostra soluzione potete fare liberamente uso delle funzioni presenti nella libreria standard di C.

1 Funzionalità richieste

Il programma deve supportare due modalità: *offline* (senza interazione con l'utente) e *interattiva*. In modalità offline, il programma dovrà:

- leggere da file uno schema di Ruzzle e un vocabolario, il cui formato è descritto in Sezione 3;
- ricercare le parole del vocabolario all'interno dello schema e calcolare il punteggio massimo ottenibile per ciascuna parola;
- salvare su file il punteggio massimo calcolato per ciascuna parola e le mosse effettuate per realizzare il punteggio.

In modalità interattiva, invece, il programma dovrà:

- richiedere l'inserimento da tastiera dei dati relativi allo schema di Ruzzle: dimensione, lettere presenti e bonus associati alle caselle;
- fino a quando l'utente non chiede di uscire dal programma, dovrà leggere in input una parola e stampare a schermo tutte le possibili sequenze di mosse che possono essere fatte per comporre la parola con il relativo punteggio, ordinate per punteggio in ordine decrescente.

Il programma deve essere avviato in modalità interattiva quando da riga di comando non viene passato alcun argomento:

```
./ruzzle
```

Quando vengono passati, nell'ordine, il pathname del dizionario, dello schema e del file di output, il programma deve essere eseguito in modalità offline:

```
./ruzzle <dizionario> <schema> <output>
```

La funzione `main` può prevedere due parametri: `argv` è un vettore di stringhe contenente gli argomenti passati da riga di comando (notare che `argv[0]` contiene il nome del programma stesso), mentre `argc` contiene il numero di elementi presenti in `argv`. Per utilizzarli, definite il `main` come segue:

```
int main(int argc, char *argv[]) {
        ...
}
```

Figura 1: Schema di Ruzzle di dimensione 4×4.

2 Ruzzle

In Ruzzle è presente un campo di dimensione $n \times n$ (dove $n > 0$) contenente lettere dell'alfabeto inglese, possibilmente con ripetizione. Utilizzando le lettere presenti all'interno dello schema, il giocatore deve trovare il maggior numero possibile di parole rispettando i seguenti vincoli:

- ciascuna lettera è adiacente alla successiva in orizzontale, verticale o diagonale (senza wrap-around);
- ciascuna posizione dello schema può essere attraversata al più una volta per comporre una parola.

Consideriamo lo schema in Figura 1:

- la parola *PASTA* può essere composta con le lettere in posizione $(3,3) \to (2,2) \to (1,2) \to (1,3) \to (0,2)$;
- la parola *ESATTA* può essere composta con le lettere in posizione $(0,3) \to (1,2) \to (0,2) \to (1,3) \to (2,3) \to (2,2)$;
- la parola *RATTI* **non** può essere composta con la sequenza $(0,2) \to (2,2) \to (1,3) \to (2,3) \to (3,2)$, poiché la posizione $(0,2)$ non è adiacente alla posizione $(2,2)$;
- la parola *RETTA* **non** può essere composta con la sequenza $(1,0) \to (0,3) \to (1,3) \to (2,3) \to (2,2)$, dal momento che, avendo escluso il wrap-around, la posizione $(1,0)$ non è adiacente alla posizione $(0,3)$;
- la parola *TESTATA* **non** può essere composta con la sequenza $(1,3) \to (0,3) \to (1,2) \to (2,3) \to (2,2) \to (1,3) \to (0,2)$, poiché la posizione $(1,3)$ è stata utilizzata due volte per comporre la parola.

A ciascuna parola è associato un punteggio che dipende dai punti assegnati alle lettere e dai bonus presenti nelle caselle utilizzate. I punti base attribuiti a ciascuna lettera sono riportati in Tabella 1.

Lettere	Punteggio
J, Q, W, X	10
G, H, Z	8
B, D, F, K, P, V	5
Y	4
L, M, N, U	3
C, R, S, T	2
A, E, I, O	1

Tabella 1: Punteggi delle lettere.

Ad alcune caselle dello schema può essere attribuito uno tra i seguenti bonus:

- *Double Letter* (DL): il punteggio base attribuito alla lettera presente nella casella è duplicato;
- *Triple Letter* (TL): il punteggio base attribuito alla lettera presente nella casella è triplicato;
- *Double Word* (DW): il punteggio attribuito alla parola è duplicato;
- *Triple Word* (TW): il punteggio attribuito alla parola è triplicato.

Nel calcolo del punteggio devono essere applicati prima i bonus relativi alle lettere, poi quelli relativi alle parole. I bonus sono cumulabili: ad esempio, nel caso siano state usate due lettere con bonus DW e TW, allora il punteggio totale della parola deve essere moltiplicato per 6. Seguono alcuni esempi relativi allo schema in Figura 1:

- alla parola *ZERO*, composta con le lettere in posizione $(0,1) \to (1,1) \to (1,0) \to (0,0)$, è attribuito il punteggio $12 = 8 + 1 + 2 + 1$;
- alla parola *TRE*, composta con le lettere in posizione $(3,0) \to (2,0) \to (1,1)$, è attribuito il punteggio $10 = (2 + 2 + 1) \cdot 2$;
- alla parola *PASTI*, composta con le lettere in posizione $(3,3) \to (2,2) \to (1,2) \to (2,3) \to (3,2)$, è attribuito il punteggio $34 = (5{+}1{+}2{\cdot}3{+}2{\cdot}2{+}1){\cdot}2$;
- alla parola *PASTA*, composta con le lettere in posizione $(3,3) \to (2,2) \to (1,2) \to (1,3) \to (0,2)$, è attribuito il punteggio $102 = (5 + 1 + 2 \cdot 3 + 2 + 1 \cdot 3) \cdot 2 \cdot 3$;
- alla parola *ESATTA*, composta con le lettere in posizione $(0,3) \to (1,2) \to (0,2) \to (1,3) \to (2,3) \to (2,2)$, è attribuito il punteggio $102 = (1 + 2 \cdot 3 + 1 \cdot 3 + 2 + 2 \cdot 2 + 1) \cdot 3 \cdot 2$.

3 File di input

Il programma dovrà leggere due file, lo schema di Ruzzle e il dizionario, il cui formato è descritto in questa sezione. In tutti i file che saranno utilizzati per il test possono essere presenti alcune righe vuote che vanno ignorate. Inoltre viene utilizzata la convenzione Unix dove il ritorno a capo è codificato dal carattere `\n`:

su Windows, invece, il ritorno a capo è codificato dalla coppia di caratteri `\r\n`, quindi se create dei file di test su questa piattaforma assicuratevi di configurare in maniera appropriata il vostro editor di testo.

3.1 Schema di Ruzzle

Il file inizia con una riga contenente un numero intero positivo n che rappresenta il numero di righe e colonne dello schema.[1] Seguono n righe di n caratteri ciascuna contenenti le lettere (in maiuscolo) che compongono lo schema. Infine sono presenti n righe di n caratteri ciascuna che indicano i bonus presenti nelle caselle: le lettere `d`, `t`, `D`, `T` rappresentano, rispettivamente, i bonus DL, TL, DW, TW mentre il carattere `.` indica l'assenza di bonus nella casella. Un possibile file che rappresenta lo schema in Figura 1 è mostrato nel Codice 1.

Codice 1: File di input per lo schema in Figura 1.

```
4

OZAE
REST
RUAT
TIIP

..t.
..tT
..Dd
D...
```

3.2 Dizionario

Il file contiene, in ciascuna riga, una parola composta esclusivamente da lettere, maiuscole o minuscole. Ai fini della ricerca di una parola, il fatto che una lettera sia maiuscola o minuscola non è rilevante. Potete assumere che non siano presenti duplicati all'interno del file. Un possibile dizionario è mostrato nel Codice 2.

Codice 2: Esempio di dizionario.

```
Pasta
esatta
raTTi
zeRo
```

[1] Esistono varie tecniche per l'allocazione dinamica di array multidimensionali: consultate la pagina `http://c-faq.com/aryptr/dynmuldimary.html` per maggiori dettagli, ma ignorate la parte relativa allo standard C99.

4 File di output

Quando avviato in modalità offline, il programma dovrà produrre in output un file dove, per ogni parola del dizionario trovata all'interno dello schema, è presente una riga in cui vengono riportati la parola, il punteggio e la sequenza di mosse nel formato `(R, C)->(R, C)->...->(R, C)`, dove `R` e `C` sono gli indici di riga e colonna delle caselle attraversate. Se per una certa parola esistono più sequenze di mosse che permettono di ottenere il punteggio massimo, si scriva nel file solo una di esse. Le parole devono occorrere nel file di output nello stesso ordine in cui appaiono nel dizionario. Ad esempio, dato il dizionario in Codice 2 e lo schema in Figura 1, il file di output è il seguente:

Codice 3: Esempio di file di output.

```
Pasta 102 (3,3)->(2,2)->(1,2)->(1,3)->(0,2)
esatta 102 (0, 3)->(1, 2)->(0,2)->(1,3)->(2,3)->(2,2)
zeRo 12 (0,1)->(1,1)->(1,0)->(0,0)
```

5 Elementi di valutazione

Sebbene il progetto sia un lavoro che può essere svolto in gruppo, la valutazione sarà individuale. Oltre alla discussione orale, saranno valutate:

- qualità del codice: oltre alla correttezza, saranno considerate l'eleganza della soluzione implementata, l'utilizzo di indentazione corretta e consistente, la suddivisione in sottoprogrammi, etc.;
- qualità della documentazione: codice non commentato non sarà valutato;
- eventuali funzionalità extra.[2]

6 Funzionalità extra

Siete liberi di implementare altre funzionalità, ad esempio la gestione degli errori nei file di input o l'implementazione di un'interfaccia grafica particolare per la modalità interattiva usando librerie come `ncurses`. Non modificate le regole di base del gioco e per il calcolo dei punteggi (almeno per la modalità offline), dal momento che verrà eseguito un test automatico preliminare sulla correttezza degli output forniti dal programma. Proposte particolarmente interessanti o eleganti potrebbero convincere il Prof. Marin a darvi un bonus all'esame :)

[2] Non sono necessarie per ottenere il punteggio massimo se gli altri requisiti sono soddisfatti.