# Università degli Studi di Napoli Federico II

Corso di Laurea in Informatica

## Laboratorio di Programmazione (Gr.3)

## Compito del 20/06/2022

**2 ore** **Dott. Andrea Apicella**

---

### REGOLE

- *Della libreria standard del C, è accettato l'utilizzo **solo** dei seguenti file header:*
  - *stdio.h*
  - *stdlib.h*
  - *math.h*
  - *string.h*
  - *time.h*

  *L'utilizzo di qualsiasi altro file header (e delle relative funzioni) della libreria standard **non sarà accettato**;*
- *Seguire scrupolosamente le direttive nella traccia ed utilizzare eventuali nomi di variabili e/o funzioni dati senza variazioni (anche in termini di maiuscole/minuscole e caratteri _);*
- *Ogni file sorgente dovrà contenere nelle prime righe un commento nel formato:*

```
/* MATRICOLA: ...
COGNOME: ...
NOME: ... */
```

- *E' fortemente consigliato commentare il codice il più possibile;*
- *E' fortemente consigliato modulare il progetto su più file, nello specifico almeno 3:*
  1. *contenente i prototipi e le definizioni di eventuali strutture;*
  2. *contenente la definizione della funzione main(). Nel caso siano richiesti più main(), fare un file diverso per ognuno di essi;*
  3. *contenente le definizioni delle funzioni rimanenti.*

  *Se lo si ritiene opportuno, è possibile separare i sorgenti in più file, motivandolo opportunamente con dei commenti;*
- ***Non** includere il file eseguibile ed eventuali file oggetto nella consegna.*
- *La consegna finale dovrà quindi contenere (almeno) 4 file:*
  - *i file sorgenti (almeno 3, come specificato sopra);*
  - *il file 'istruzioni.txt', contenente i comandi per la compilazione/linking come richiesti nella traccia.*

---

# TRACCIA DEL 20/06/2022

*TIC TAC TOE* (meglio conosciuto in Italia come *Tris*) è un gioco a due giocatori che prevede l'inserimento di un simbolo (di solito una $X$ o un $O$, in base al giocatore) in una griglia $3 \times 3$. Vince il giocatore che, per primo, riesce a disporre una sequenza di tre ripetizioni consecutive del suo simbolo lungo una riga, o lungo una colonna, o lungo una diagonale della griglia di gioco. Nel caso in cui la griglia venga riempita totalmente senza ottenere una configurazione di vittoria, la partita finirà in pareggio. Esempi:

configurazioni di vittoria per il giocatore $X$:

$$\begin{bmatrix} X & O & O \\ X & O & X \\ X & X & O \end{bmatrix}, \begin{bmatrix} X & O & O \\ O & X & \\ & & X \end{bmatrix}, \begin{bmatrix} O & O & X \\ O & O & \\ X & X & X \end{bmatrix}$$

configurazioni di vittoria per il giocatore $O$ :

$$\begin{bmatrix} X & O & O \\ X & O & X \\ O & X & O \end{bmatrix}, \begin{bmatrix} O & O & O \\ X & O & X \\ & & O \end{bmatrix}, \begin{bmatrix} O & O & \\ X & O & X \\ & O & \end{bmatrix}$$

configurazione di pareggio:

$$\begin{bmatrix} X & O & O \\ O & O & X \\ X & X & O \end{bmatrix}, \begin{bmatrix} X & O & O \\ O & X & X \\ O & X & O \end{bmatrix}, \begin{bmatrix} O & X & O \\ O & X & X \\ X & O & X \end{bmatrix}$$

Data una configurazione

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix},$$

è possibile rappresentare quest'ultima attraverso un'unica sequenza di 9 elementi

$$c_{11}c_{12}c_{13}c_{21}c_{22}c_{23}c_{31}c_{32}c_{33}$$

con $c_{ij} = \begin{cases} X & \textit{se l'elemento in posizione i,j è una } X \\ O & \textit{se l'elemento in posizione i,j è una } O \\ @ & \textit{se la posizione i,j è vuota.} \end{cases}$

Esempio: la sequenza $X@OOOXXXO$ corrisponde alla configurazione:

$$\begin{bmatrix} X & & O \\ O & O & X \\ X & X & O \end{bmatrix}$$

Un file di testo 'configurazioni.txt' contiene, su ogni riga, una possibile configurazione del campo da gioco.

Esempio:

'configurazioni.txt':

```
XOOX@XXXO
XOOXOXOXO
XOOOOXX@O
```

contiene le configurazioni:

$$\begin{bmatrix} X & O & O \\ X & & X \\ X & X & O \end{bmatrix}, \begin{bmatrix} X & O & O \\ X & O & X \\ O & X & O \end{bmatrix}, \begin{bmatrix} X & O & O \\ O & O & X \\ X & & O \end{bmatrix}.$$

Si chiede di scrivere un programma in linguaggio C che effettui le seguenti operazioni:

**Punto 1:** caricare il contenuto del filein una lista semplicemente concatenata. La funzione di caricamento dovrà chiamarsi *load_matches_from_file(...)*. Ogni nodo della lista dovrà contenere:

- la configurazione della partita, disposta in una matrice *campo* di $3 \times 3$ caratteri;
- una variabile *vincitore* di tipo 'char', che dovrà contenere uno dei seguenti valori:
  - 'X', se la configurazione è di vittoria per il giocatore 'X';
  - 'O', se la configurazione è di vittoria per il giocatore 'O';
  - 'P', se la configurazione è di pareggio.

  il vincitore dovrà essere determinato dall'invocazione di un'apposita funzione *the_winner_is(...)* avente come unico parametro un campo da gioco.
- un variabile *punteggio*, di tipo 'int', che dovrà contenere il punteggio della partita calcolato nel seguente modo:

$$\text{punteggio} = \begin{cases} 0 & \text{se la partita è un pareggio} \\ 3 + \text{numero posizioni libere} & \text{altrimenti.} \end{cases}$$

il punteggio dovrà essere calcolato da una apposita funzione *match_score(...)* che dovrà restituire un 'int'. Il numero di posizioni libere dovrà essere calcolato da una seconda funzione *free_positions(...)*. Quest'ultima dovrà implementare una soluzione *preferibilmente ricorsiva*;

**Punto 2:** stampare a video la lista attraverso un'apposita funzione *print_list(...)* *ricorsiva*. Tale funzione dovrà invocare al suo interno una funzione *print_match(...)* dedicata alla stampa dei dettagli di una singola partita, ossia:

- il vincitore $X$ o $O$ (o l'eventuale pareggio);
- il punteggio fatto;
- il campo da gioco.

**Punto 3:** rimuovere dalla lista tutte le configurazioni di pareggio attraverso una apposita funzione *remove_tied(...)*. Stampare a video la lista degli elementi residui;

**Punto 4:** salvare in due file di testo differenti 'vittorie_X.txt' e 'vittorie_O.txt' rispettivamente le configurazioni di vittoria di 'X' e di 'O' ancora presenti nella lista, nello stesso formato del file di input. Il salvataggio dovrà essere effettuato attraverso la *doppia* invocazione di un'*unica* funzione *write_winners_on_file(...)* avente come parametri:

- il nome del file;
- la lista;
- un carattere che indica il giocatore di cui si desiderano salvare le configurazioni vincenti.

la funzione dovrà essere *non* distruttiva della struttura dati;

**Punto 5:** compilare ed eseguire il programma da linea di comando. Riportare su di un file di testo di nome 'istruzioni.txt' i comandi necessari per effettuare la compilazione con *gcc* dei *singoli* moduli ed il linking. Compilazione e linking dovranno essere effettuati *separatamente*. Il file eseguibile dovrà avere nome **prog**_numerodimatricola**.eseguibile**.