Il compito principale del programma è quello di confrontare due parole di ugual lunghezza.

Simboli ammessi 🡪 le stringhe possono essere composte da:

* Caratteri alfabetici maiuscoli e minuscoli (**A-Z** e **a-z**);
* Cifre numeriche (**0-9**);
* I simboli ‘**–**‘ e ‘**\_**’.

Ordine degli eventi🡪 il sistema:

* **Legge come parametri in ingresso:**
  + **Int k**: lunghezza delle stringhe;
  + **Char [] okWords**: insieme delle stringhe ammissibili.
* **Legge da standard input (tastiera):**
  + **+inserisci\_inizio**: indica il fatto che da adesso in poi verranno inserite delle stringhe che dovranno essere aggiunte ad okWords.

**Nota**: si da per scontato che non si inseriscano parole duplicate.

* + **+inserisci\_fine**: indica che la sequenza di parole da aggiungere è finita.

**Nota**: si da per scontato che non ci siano parole duplicate

* + **+nuova\_partita** : indica l’inizio di una nuova partita (andranno effettuate le opportune operazioni per scegliere la parola da indovinare ecc.).
  + **Char [] attemp**: è il tentativo di indovinare la parola da parte del giocatore (sempre di lunghezza k).

Nota: è possibile fare **int n** tentativi.

* + **+stampa\_filtrate**: stampa tutte le parole del dizionario che sono compatibili con “ciò che si è imparato sulla parola da indovinare” fino a quel momento. Si escludono quindi:
    - Parole in cui ci sono simboli ormai esclusi;
    - Parole in cui un simbolo compare in un posto dove non può farlo;
    - Parole in cui un simbolo compare meno volte del numero minimo “scoperto tramite i tentativi”;
    - Se si è ormai scoperto il numero di volte in cui un simbolo deve apparire in una parola, le parole che contengono quel simbolo un numero diverso di volte da quello atteso.

Tali parole vanno stampate in ordine lessicografico (ordine dell’ASCII).

Ogni volta che si effettua un confronto, dopo andranno stampate queste parole come se si fosse ricevuto il comando “+stampa\_filtrata”.

* **Scrive in output:**
  + Quante lettere sono **giuste al posto giusto** (**+**);
  + Quante lettere sono **giuste al posto sbagliato** (**|**);
  + Quante lettere sono **sbagliate al posto sbagliato** (**/**).

**Nota🡪** nel confronto si ha che i giusti al posto giusto prendono priorità sui giusti al posto sbagliato. Ciò vuol dire che se nella stringa ho le parole “casa” e “caaa”, si ha che la sequenza di lettere ammessa sarà ✓✓x✓.

* **La partita finisce se:**
  + **Dopo n tentativi non si è ancora indovinata la parola:** si stampa “KO.” E la partita termina.
  + **Entro gli n tentativi si indovina la parola:** si stampa “OK.” e la partita termina.

Comando per compilare: $gcc -Wall -Werror -02 -g3 sorgente.c -0 “eseguibile”

Per confrontare i risultati dei test ottenuti con quelli attesi:

* $diff ./public\_output ./program\_output

Se non viene dato nessun output vuol dire che sono identici.

Command line:

* Valgrind 🡪 suite di strumenti per l’analisi dinamica dei programmi.
  + Memcheck: rivela errori nell’uso della memoria dinamica:
    - * De-allocazioni mancanti (memory leaks);
      * Use after free;
      * Doppie free.
  + Callgrind: tracciamento tempo macchina per funzione (si usa per trovare i colli di bottiglia).
  + Massif: tracciamento quantità memoria dinamica.
* Address Sanitizer🡪 serve per intercettare errori di accesso alla memoria:
  + - * Acceso oltre i limiti di un array;
      * Accesso oltre i limiti della memoria allocata;
      * Accesso a un null pointer.

Valgrind e Adress Sanitizer non possono essere usati insieme. Tenere attivo Address Sanitizer e disattivarlo solo quando si deve usare Valgrind.

Contatti tutor:

* Paolo Daolio: [paolo.daolio@mail.polimi.it](mailto:paolo.daolio@mail.polimi.it)
* Susanna Sidoti: [susanna.sidoti@mail.polimi.it](mailto:susanna.sidoti@mail.polimi.it)