Laboratorio per il corso di Reti – Applicazione A

Sviluppare in linguaggio C un'applicazione di rete, costituita da un programma server ed un programma client, in cui l'utente del programma client cerca di indovinare, tramite una serie di indizi, una parola di cinque lettere scelta dal programma server, in maniera simile alla popolare applicazione Wordle (https://www.nytimes.com/games/wordle/index.html).
Il comportamento specifico è comunque molto più semplice di quello di Wordle, qui citato solo come riferimento, e viene descritto in dettaglio nel seguito.

SERVER

1. Il programma server viene eseguito indicando sulla riga di comando la porta sulla quale mettersi in ascolto
   `$ programma_server <numero porta> [<max tentativi>]`
2. Il numero massimo di tentativi è opzionale e può assumere un valore tra 6 e 10. Se non viene indicato sulla riga di comando, il valore di default è 6.
3. Il server definisce internamente due o più parole di cinque lettere, tra le quali scegliere in maniera casuale quella da indovinare. Le parole devono essere di senso compiuto e costituite da caratteri alfabetici.
4. La parola da identificare ("*target*") è fissata per l'intera connessione tra server e client, ma può cambiare tra una connessione e l'altra.
5. I messaggi inviati dal server hanno una lunghezza massima di 256 caratteri e devono essere terminati da un carattere di andata a capo ('`\n`'). Diversi campi dei messaggi sono separati da un singolo spazio (carattere ASCII 32, 0x20).
6. All'apertura della connessione il server manda un messaggio di benvenuto, nel formato
   `OK <Max Tentativi> <Messaggio>`
   ovvero la stringa '`OK`', seguita da uno spazio, dal numero massimo di parole accettate, uno spazio, e da una stringa personalizzabile dal server.
7. Il server si pone in attesa di un messaggio di *comando* da parte del client.
8. I possibili comandi sono:

Comando	**Descrizione**
WORD	Tentativo di identificazione
QUIT	Termina la comunicazione

9. I comandi sono costituiti da stringhe della lunghezza massima di 256 caratteri, terminate da un carattere di andata a capo ('`\n`').
10. La semantica e il formato dei comandi sono i seguenti:
    a. Il comando WORD permette al client di indicare la parola che cerca di identificare, e ha il formato
       `WORD <testo>`
       ovvero la stringa '`WORD`', seguita da uno spazio e dalla parola di cinque lettere che si vuole identificare ("*guess*"), e terminata da un carattere di andata a capo ('`\n`').
       Ad esempio:
       `WORD cesto`
       La parola che si vuole identificare DEVE essere lunga cinque caratteri.
    b. Il comando QUIT permette al client di chiudere la connessione prima che la parola sia stata indovinata e ha il formato

`QUIT`
ovvero la stringa '`QUIT`' seguita da un carattere di andata a capo ('`\n`').

11. In tutti i comandi, il carattere di separazione tra i campi, se presente, è sempre costituito da un singolo spazio (carattere ASCII 32, 0x20). La parola *guess* da identificare deve essere costituita da caratteri ASCII (classe *ascii*) alfabetici, riconosciuti come tali dal sistema locale (classe *alpha*): qualsiasi altro carattere viene considerato illegale.
12. Il server esamina il messaggio ricevuto e ne verifica la correttezza sintattica. Se il messaggio non è sintatticamente corretto risponde con il messaggio
    `ERR <messaggio>`
    ovvero la stringa '`ERR`' seguita da uno spazio e da un messaggio di testo che descrive la natura dell'errore, e chiude la connessione.
13. Se il messaggio è sintatticamente corretto, il server risponde in base al comando ricevuto:
    a. Alla ricezione del comando WORD, il server confronta, senza distinguere tra maiuscole e minuscole, la parola indicata con la parola *target*.
       i. Se le due parole sono uguali, risponde con un messaggio di successo, nel formato
          `OK PERFECT`
          ovvero la stringa '`OK PERFECT`' seguita da un carattere di andata a capo ('`\n`'), dopo di che chiude la connessione.
       ii. Se le due parole differiscono e non sono stati esauriti i tentativi a disposizione, risponde con un messaggio che contiene un "*" per i caratteri corretti in posizione corretta, un "+" per i caratteri corretti in posizione errata, e un '-' per i caratteri non corretti in quanto non presenti nella parola da identificare, nel formato
          `OK <tentativo> <s1><s2><s3><s4><s5>`
          ovvero la stringa '`OK`' seguita da uno spazio e dall'indice del tentativo corrente, uno spazio, e i simboli che indicano quali e quanti siano i caratteri corretti, seguita da un carattere di andata a capo ('`\n`'), ad esempio
          `OK 2 *+-+*`
          indica, al secondo tentativo, che il primo e quinto carattere sono corretti in posizione corretta, il secondo e il quarto corretti ma in posizione sbagliata, e il terzo non è presente nella parola da indovinare
       iii. Se le due parole differiscono ma sono stati esauriti i tentativi a disposizione, risponde con un messaggio di fallimento, nel formato
          `END <tentativo> <parola_target>`
          ovvero la stringa 'END' seguita da uno spazio e dall'indice del tentativo corrente, uno spazio, e la parola target che si sarebbe dovuta indovinare, seguiti da un carattere di andata a capo ('`\n`'), dopo di che chiude la connessione. Ad esempio:
          `END 10 vasto`
    b. Alla ricezione del comando QUIT il server risponde con un messaggio di chiusura nel formato
       QUIT <messaggio>
       ovvero la stringa '`QUIT`' seguita da uno spazio e da un messaggio di commiato terminato da un carattere di andata a capo ('`\n`'), dopo di che chiude la connessione. Il messaggio di commiato può contenere o meno la parola che si sarebbe dovuta identificare.

14. Dopo aver elaborato i dati e mandato il messaggio finale (casi 13.a.i, 13.a.iii e 13.b), o dopo aver mandato qualsiasi messaggio di errore, il server chiude la connessione e si pone in attesa della richiesta di un nuovo client.

CLIENT

1. Il programma client viene eseguito indicando sulla riga di comando l'indirizzo IPv4 del server da contattare e la porta sulla quale contattarlo
   `$ programma_client <indirizzo_server> <numero_porta>`
2. Il client elabora sempre i messaggi che riceve dal server, ovvero non ne presenta il contenuto direttamente all'utente, ma rimuove qualsiasi delimitatore del protocollo e mostra all'utente l'informazione ricevuta in maniera chiara e contestualizzata, evitando qualsiasi diagnostica non rilevante.
3. Dopo l'apertura della connessione il client si aspetta di ricevere dal server il messaggio di benvenuto, nel formato
   `OK <Max Tentativi> <Messaggio>`
   ovvero la stringa '`OK`', seguita da uno spazio, dal numero massimo di tentativi a disposizione, uno spazio, e da una stringa personalizzabile dal server. Il messaggio può avere una lunghezza massima di 256 caratteri ed è terminato da un carattere di andata a capo ('`\n`')
4. Il client presenta all'utente il messaggio del server, senza il delimitatore "`OK `" (che costituisce una parola chiave del protocollo di scambio) e il numero massimo di tentativi, che non fanno parte del messaggio del server.
5. Il client a questo punto invita l'utente ad indovinare la parola misteriosa, tenendolo aggiornato riguardo ai tentativi rimanenti a disposizione e permettendogli, eventualmente, di abbandonare l'interazione con il server. I casi possibili sono:
   a. L'utente vuole fare un tentativo di identificazione
   b. L'utente vuole abbandonare l'esecuzione
6. Nel caso in cui l'utente voglia fare un tentativo (caso 5a),
   a. il client sollecita l'utente ad inserire la parola di cinque lettere, verificandone la congruenza, per inviare al server il comando
      `WORD <parola>`
      ovvero la stringa '`WORD`', seguita da uno spazio e dalla parola da indovinare, terminata da carattere di andata a capo ('`\n`'). La parola deve rispettare i requisiti indicati al punto 11 del server.
   b. Dopo aver trasmesso il comando, il client si pone in attesa della risposta del server.
      i. In caso di risposta positiva (OK PERFECT), il client offre opportunamente riscontro del successo all'utente, e chiude la connessione
      ii. Nel caso in cui la parola non sia stata ancora identificata ma i tentativi non solo ancora esauriti (OK <tentativo>), il client presenta all'utente l'esito in maniera chiara ed informativa, invitandolo ad effettuare il tentativo successivo e tornando al punto 5.
      iii. Nel caso in cui i tentativi siano stati esauriti (END), il client comunica l'esito all'utente in maniera chiara e informativa, e chiude la connessione
      iv. In caso di errore (ERR), il client riporta all'utente l'eventuale messaggio del server (senza i delimitatori del protocollo), e chiude la connessione
7. Nel caso in cui l'utente voglia abbandonare l'esecuzione (caso 5b)

    a. Il client manda al server il comando `QUIT` secondo il formato descritto al punto 10.b del server
    b. Dopo aver trasmesso il comando, il client si pone in attesa della risposta del server.
        i. In caso di risposta positiva (QUIT) secondo il formato descritto al punto 13.b del server, il client ne riporta il contenuto (senza i delimitatori del protocollo) all'utente, e chiude la connessione.
        ii. In caso di errore (ERR), il client riporta all'utente l'eventuale messaggio del server (senza i delimitatori del protocollo), e chiude la connessione
8. In ogni caso di errore e dopo la chiusura della connessione, il client termina l'esecuzione.

Entrambi i programmi devono gestire opportunamente tutti i possibili errori che si possono verificare in fase di apertura e chiusura delle connessioni. Entrambi i programmi devono operare in maniera tale da rispettare il protocollo previsto e reagire opportunamente a violazioni di tale protocollo.
Tutti i messaggi scambiati sono costituiti da stringhe terminate da un carattere di andata a capo ('`\n`'). Tutte le parole chiave sono rappresentate da caratteri maiuscoli.
Entrambi i programmi client e server devono essere in grado in inter-operare con programmi analoghi che rispettino i requisiti elencati: per questo motivo i messaggi di rete devono rispettare rigorosamente i formati definiti. L'interoperabilità è un requisito fondamentale che verrà verificato in fase di valutazione. Ogni altro aspetto di interazione tra il client, il server e l'utente, può essere personalizzato a piacimento, ad esempio i messaggi a schermo di notifica o di errore, nei limiti dell'utilizzabilità del programma stesso.
La conoscenza di queste specifiche non deve essere un requisito per l'usabilità dei programmi.

Esempi di comunicazione sulla rete

```
S: OK 6 Indovina la parola del giorno
C: WORD astio
S: OK 1 *---+
C: WORD morta
S: OK 2 +++-+
C: WORD amore
S: OK PERFECT

S: OK 6 Indovina la parola del giorno
C: WORD mossi
S: OK 1 *----
C: QUIT
S: QUIT Vai via cosi' presto? La parola era 'marea'


S: OK 6 Indovina la parola del giorno
C: WORD questo
S: ERR Parola non di 5 caratteri
```