UNIVERSITÀ DI PISA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
SPECIFICHE PROGETTO A.A. 2019/2020

Reti informatiche, cod. 545II, 9 CFU

Prof. Giuseppe Anastasi, Ing. Francesco Pistolesi

Si richiede di progettare e implementare un'applicazione distribuita client-server che riproduca il gioco del Lotto. L'applicazione deve permettere a vari host client (giocatori) di effettuare giocate e inviarle a un server remoto, il cui compito è memorizzare le giocate, effettuare le estrazioni ed elaborare le vincite, notificandone l'importo ai giocatori.

1. INTRODUZIONE

Il gioco del Lotto (o semplicemente Lotto) consiste nell'estrazione di cinque numeri tra 1 e 90, con premio per coloro che ne indovinano almeno uno.

È un gioco d'azzardo gestito dall'Ispettorato Generale per il Lotto e le Lotterie presso il Ministero delle Finanze. La raccolta delle giocate e i pagamenti delle vincite sono affidate in concessione a Lottomatica, un operatore di lotterie e scommesse italiano.

Il Lotto consiste in tre *estrazioni* ogni settimana (martedì, giovedì e sabato). Ogni estrazione coinvolge undici *ruote*: Bari, Cagliari, Firenze, Genova, Milano, Napoli, Palermo, Roma, Torino, Venezia e Nazionale. In ogni estrazione, sono estratti cinque *numeri* tra 1 e 90 per ciascuna ruota, senza reinserimento.

Il gioco consiste nello scommettere sui numeri che saranno estratti sulle varie ruote. L'ordine in cui sono estratti non è significativo. Si può scommettere sull'uscita di uno o più numeri su una ruota, su più ruote o su tutte le ruote. In particolare, data una ruota, si parla di *tipo di giocata*:

- *estratto*, se si scommette sull'uscita di un solo numero;
- *ambo*, se si scommette sull'uscita contestuale di due numeri;
- *terno*, se si scommette sull'uscita contestuale di tre numeri;
- *quaterna*, se si scommette sull'uscita contestuale di quattro numeri;
- *cinquina*, se si scommette sull'uscita contestuale di cinque numeri.

La giocata si effettua grazie a una schedina come quella mostrata in Fig. 1, sulla quale possono essere scelti fino a n=10 numeri. L'importo della vincita è fisso e dipende da quanti numeri sono stati indovinati, da cosa si è giocato (ambo, terno, eccetera), e da quanti numeri sono stati complessivamente giocati nella schedina.

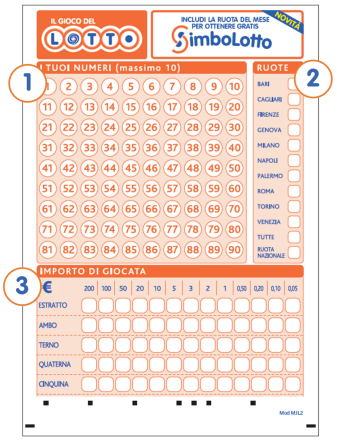

Figura 1 Una tipica schedina per il gioco del Lotto.

Se nella schedina si giocano più di 5 numeri, c'è una maggiore probabilità di vincita perché combinando gli *n* numeri giocati si possono ottenere più estratti, ambi, terni e così via. Per esempio, consideriamo una schedina in cui il giocatore A gioca i numeri 5, 18 e 33. Se, come tipo di giocata, A sceglie di puntare 1 € sul terno, l'importo giocato sarà tutto giocato sull'unico terno possibile (cioè 5, 18, 33). Se, nell'ambito della stessa estrazione, un giocatore B giocasse una schedina con i numeri 5, 18, 33 e 59, e scegliesse anch'egli di puntare 1 € sul terno, stavolta l'importo di 1 € sarebbe equamente ripartito in sotto-importi, ciascuno dei quali associato a ciascun terno che si può generare dai quattro numeri giocati (cioè 5, 18, 33; 5, 18, 59; 18, 33, 59; e così via). È ovvio che se poi, durante l'estrazione, escono i numeri 5, 18 e 33, il giocatore A riceverà un importo di vincita maggiore perché, come si dice in gergo, ha puntato 1 € sul terno "secco". Il giocatore B riceverà invece una vincita proporzionale al sotto-importo. Per ogni Euro giocato, il valore della vincita relativo a ogni tipologia di giocata è fisso, ed è mostrato nella Tabella 1.

Tabella 1 Vincita per ogni Euro giocato, per ogni tipologia di giocata

Numeri giocati	Numeri indovinati	Vincita (€)
1	1	11.23
2	2	250
3	3	4500
4	4	120000
5	5	6000000

Nella stessa schedina, è possibile abbinare più tipi di giocata. Un giocatore C potrebbe ad esempio giocare i numeri 5, 18 e 33, puntando 1 € sul terno e 2 € sull'ambo. Se fra i cinque numeri estratti esce il terno, il giocatore vince 4500 €. Se esce un ambo, il giocatore vince un importo pari a $250/k$ €, dove k è il numero di ambi generabili dai numeri 5, 18, e 33. Se escono più ambi, il precedente importo è moltiplicato per il numero di ambi usciti.

Se nella stessa schedina si indicano più ruote, le considerazioni precedenti continuano a valere, ma gli importi giocati sono equamente ripartiti sulle ruote scelte. Quindi, per esempio, il giocatore C potrebbe fare la stessa giocata descritta sopra, ma stavolta indicare due ruote: Roma e Firenze. In questo caso, le vincite sarebbero divise per due, quindi il terno uscito su una delle due ruote è pagato 2250 €, mentre gli ambi sono pagati ciascuno $250/2k$ €. Infine, se il giocatore C effettua la giocata su tutte le ruote, le vincite sono divise per 11, numero totale di ruote. Quindi, se esce il terno su Napoli e due ambi (per esempio uno su Bari e l'altro sulla ruota Nazionale) il giocatore C vince 4500/11 € per il terno, più $250/11k$ € per ciascuno dei due ambi.

2. IMPLEMENTAZIONE

L'applicazione distribuita da sviluppare sarà basata sul **paradigma client-server** è dovrà implementare quanto descritto nel Paragrafo 1. In particolare, il server deve consentire ai client di fare giocate per la prossima estrazione prevista, visualizzare le ultime giocate e le ultime vincite. Il server dovrà essere in grado di **gestire richieste concorrenti**. D'altra parte, il client deve permettere di inviare le giocate e utilizzare le altre funzionalità descritte nel seguito.

2.1 SERVER

Il server **lotto_server** riceve le richieste dai vari client e le serve, eseguendo il comando ricevuto. Il server deve gestire la concorrenza tramite l'**approccio multi-processo**. Il server è mandato in esecuzione come segue:

```
./lotto_server <porta> <periodo>
```

dove:

<porta> è la porta su cui il server è in ascolto. Il server dovrà essere raggiungibile da tutti gli indirizzi IP della macchina su cui è in esecuzione;

<periodo> parametro opzionale che specifica il tempo fra un'estrazione e la successiva. Se non si specifica il parametro, le estrazioni sono effettuate ogni 5 minuti.

Una volta eseguito, **lotto_server** mostra a video le informazioni sullo stato (creazione socket di ascolto, connessioni ricevute, operazioni richieste dai client ecc.), con un formato a piacere. Per ogni client, il server deve memorizzare su un file locale le informazioni sulle giocate e sulle vincite, in modo da poterle inviare al client quando richieste.

2.1.1 DETTAGLIO DEI COMANDI

I comandi accettati incondizionatamente dal server sono:

`!signup username password`
registra un nuovo utente caratterizzato da username e password ricevuti come parametri. Prima di farlo, il server verifica che non esista già un utente con lo stesso username. Se c'è, invia un messaggio di errore al client, attendendo uno username diverso, ricevuto il quale ripeterà il controllo di unicità. Questi step sono ripetuti finché il client non inserisce uno username non presente. Quando il comando va a buon fine, il server crea un *file registro* relativo al nuovo utente, dove conseguentemente memorizzerà le giocate e le vincite realizzate. Invia infine un messaggio al client per dare conferma dell'avvenuta registrazione.

`!login username password`
autentica un utente permettendo di salvare le giocate e le vincite realizzate nel relativo file registro. Se l'utente ha specificato delle credenziali valide, il server genera una stringa di 10 caratteri alfanumerici casuali (*session id*) e la mantiene in memoria, associandola allo username autenticato. Invia successivamente il session id al client, che lo memorizza. Se l'autenticazione con nome utente e password fallisce, il server deve mettere a disposizione del client due ulteriori tentativi, falliti i quali deve chiudere la connessione. Il server salva in un file di testo l'indirizzo IP del client e il timestamp in cui si è verificato il fallimento del terzo tentativo. Il server deve bloccare l'indirizzo IP per 30 minuti. Se lo stesso client tenta nuovamente di connettersi in questo intervallo di tempo, deve ricevere un errore che spieghi la ragione del blocco.

Dopo il login, il server accetta i comandi per le giocate descritti di seguito, solo se inviati tramite messaggi contenenti un session id valido:

!invia_giocata schedina
riceve una nuova schedina dall'utente. Una volta ricevuto il messaggio contenente la schedina, il server deve opportunamente decodificarla e memorizzarla nel file registro, per renderla definitiva. Una volta fatto ciò, notifica il client dell'avvenuta giocata.

!vedi_giocate tipo
invia al client le informazioni sulle giocate effettuate. Se il parametro **tipo** vale 0, il server invia le giocate del client relative a estrazioni già effettuate; se il parametro **tipo** vale 1, il server invia le giocate *attive*, cioè quelle in attesa della prossima estrazione.

!vedi_estrazione n ruota
invia al client i numeri estratti nelle ultime n estrazioni, sulla ruota ricevuta. Se la ruota non è stata specificata, il server invia le informazioni di tutte le ruote.

!vedi_vincite
legge le vincite del cliente nel file registro e ne invia un consuntivo al cliente. Per ogni vincita, il server invia l'estrazione in cui è stata realizzata e il totale per tipologia di giocata.

!esci
il server invalida il session id, chiude il/i file, invia un messaggio al client di avvenuto logout, chiude il socket TCP ed esce. Il server stampa un messaggio che documenta la disconnessione del client. Il server dovrà gestire appropriatamente la disconnessione e, in caso di errore nel completare la procedura, deve inviare un opportuno messaggio al client.

2.2 CLIENT

Il client si avvia con la seguente sintassi:

./lotto_client <IP server> <porta server>

dove:

<IP server> è l'indirizzo dell'host su cui è in esecuzione il server;
<porta server> è la porta su cui il server è in ascolto;

All'avvio, i comandi disponibili sono:

- **!help <comando>**
- **!signup <username> <password>**
- **!login <username> <password>**
- **!invia_giocata <schedina>**
- **!vedi_giocate <tipo>**
- **!vedi_estrazione <n> <ruota>**
- **!vedi_vincite**
- **!esci**

Il client deve mostrare messaggi relativi agli errori che si verificano durante l'esecuzione. I messaggi devono spiegare brevemente la natura dell'errore. Evitare messaggi generici. Un esempio di esecuzione è il seguente:

```
$ ./lotto_client 127.0.0.1 4242


*****************************GIOCO DEL LOTTO *****************************
Sono disponibili i seguenti comandi:

1) !help <comando> --> mostra i dettagli di un comando
2) !signup <username> <password> --> crea un nuovo utente
3) !login <username> <password> --> autentica un utente
4) !invia_giocata g --> invia una giocata g al server
5) !vedi_giocate tipo --> visualizza le giocate precedenti dove tipo = {0,1}
                          e permette di visualizzare le giocate passate '0'
                          oppure le giocate attive '1' (ancora non estratte)
6) !vedi_estrazione <n> <ruota> --> mostra i numeri delle ultime n estrazioni
                                    sulla ruota specificata
7) !esci --> termina il client
```

2.2.1 DETTAGLIO DEI COMANDI

I comandi accettati dal client sono i seguenti:

!help comando
mostra i dettagli del comando specificato come parametro. Se il comando non è specificato, **!help** restituisce una breve descrizione di tutti i comandi, come quella mostrata nella pagina precedente.

!signup username password
registra un nuovo utente caratterizzato da username e password ricevuti come parametri. Se è già presente un altro utente con lo stesso username, **!signup** deve restituire un errore al client e non deve richiedere uno username diverso. Se il comando va a buon fine, il server crea un file registro relativo al nuovo utente, dove conseguentemente memorizzerà le giocate e le vincite realizzate.

!login username password
invia al server le credenziali di autenticazione. Se sono valide, riceve un session id dal server. Il session id dovrà far parte dei messaggi relativi all'invio dei comandi che seguono, e permetterà al server di salvare le informazioni sulle giocate. Se il client invia credenziali non valide, riceve un messaggio di errore dal server.

!invia_giocata schedina
invia al server una giocata. La giocata è descritta dalla schedina, la quale contiene le ruote scelte, i numeri giocati e gli importi per ogni tipologia di giocata, come descritto nel Paragrafo 1. La schedina è formata da: elenco ruote, numeri giocati, importi per tipo giocata. La schedina si specifica come nel seguente esempio di esecuzione, nel quale il client gioca i numeri 15, 19 e 33 sulle ruote di Milano e Roma, puntando 10 € sul terno e 5 € sull'ambo:

```
> !invia_giocata -r roma milano -n 15 19 33 -i 0 5 10
```

l'opzione **-r** precede l'elenco delle ruote, l'opzione **-n** precede i numeri giocati, l'opzione **-i** precede gli importi per ogni tipologia di giocata, partendo dall'estratto fino alla cinquina. Nell'esempio precedente, lo **0** che segue l'opzione **-i** indica che non sono stati giocati gli estratti[1]. Nel seguente esempio si giocano invece i numeri 30 e 56 su tutte le ruote, puntando 1.5 € sull'ambo e 5 € sull'estratto.

[1] Significa che il client non ha scommesso sulla sola uscita del 15, del 19, o del 33. Ricordare il significato del termine *estratto* spiegato nel Paragrafo 1.

```
> !invia_giocata –r tutte –n 30 56 –i 5 1.5
```

È importante ricordare che l'importo giocato non è necessariamente un importo intero.

!vedi_giocate tipo
richiede al server le giocate effettuate. Se il parametro **tipo** vale 0, il client riceve le sue giocate relative a estrazioni già effettuate; se il parametro **tipo** vale 1, il client riceve le sue *giocate attive*, cioè quelle in attesa della prossima estrazione.

Esempio di esecuzione:

```
> !vedi_giocate 1
1) Roma 15 19 20 * 10.00 terno * 5.00 ambo
2) Milano Napoli Palermo 90 * 15.00 estratto
```

!vedi_estrazione n ruota
richiede al server i numeri estratti nelle ultime n estrazioni, sulla ruota specificata come parametro. Se la ruota non è specificata, il server invia i numeri estratti su tutte le ruote.

Esempio di esecuzione:

```
> !vedi_estrazione 1
Bari      88   18    2   45   10
Cagliari   3   38   12   59   77
Firenze   49   72   66   44   90
Genova    37   26   55   71   59
Milano     1   39   80   79   41
Napoli    19   33   45   10   70
Palermo   85   62   61   54   11
Roma      17    7   51   47   56
Torino    38    1   42   22   11
Venezia   90   33   87   12   29
Nazionale 51   69    4   54    2
```

!vedi_vincite
richiede al server tutte le vincite del client, l'estrazione in cui sono state realizzate e un consuntivo per tipologia di giocata.

Esempio di esecuzione:

```
> !vedi_vincite
Estrazione del 30-11-2019 ore 08:00
Bari  88 18    >>   Ambo 250
**********************************************
Estrazione del 11-12-2019 ore 19:35
Firenze 13 39 52   >>   Terno 0   Ambo 10
Napoli 28          >>   Estratto 22.46
**********************************************

Vincite su ESTRATTO: 22.46
Vincite su AMBO: 260.00
Vincite su TERNO: 0.00
Vincite su QUATERNA: 0.00
Vincite su CINQUINA 0.00
```

!esci
invia un messaggio di logout al server, attende la conferma dal server, chiude il socket TCP ed esce. Il server

stampa un messaggio che documenta la disconnessione del client. Il server, dovrà gestire in maniera appropriata la disconnessione di un cliente.

REQUISITI

- Client e server si scambiano dati tramite **socket TCP**. Prima di ogni scambio, il ricevente deve essere informato su **quanti byte** deve leggere dal socket. Non possono essere inviati numeri arbitrari di byte sui socket.

- Per gestire le schedine, le giocate e le vincite, è possibile utilizzare le **strutture dati a piacere**. Gli aspetti che non sono dettagliatamente specificati in questo documento possono essere implementati liberamente.

- Usare gli **autotools** (comando **make**) per la compilazione del progetto.

- Il codice **deve essere indentato e commentato** in ogni sua parte: significato delle variabili, descrizione delle strutture dati introdotte, processazioni e così via. I commenti possono essere evitati nelle parti banali del codice.

- Il **salvataggio** delle informazioni di giocate, vincite ed estrazioni sui file registro è opzionale. Nel caso si decida di non implementarlo, tali informazioni sono semplicemente mantenute in memoria.

CONSEGNA

Il progetto deve essere caricato sul sistema elearn.ing.unipi.it (sulla pagina del corso) entro 72 ore dal giorno dell'esame, usando le credenziali di Ateneo per l'accesso. Per ogni appello, sarà creata una nuova sezione per le consegne.

VALUTAZIONE

Il progetto è analizzato prima dello svolgimento dell'esame, eseguendolo su sistema operativo **Debian 8**, quella riferita durante il corso. Si consiglia di testare il codice su una macchina Debian 8 prima della consegna. Durante l'esame, sarà richiesta l'esecuzione del programma.

La valutazione del progetto prevede le seguenti fasi:

1. **Compilazione del codice**
   Il client e il server vanno compilati con opzione **`-Wall`** che mostra i vari warning. Non vi dovranno essere warning o errori. L'opzione **`-Wall`** va abilitata anche durante lo sviluppo del progetto, interpretando i messaggi forniti dal compilatore;

2. **Esecuzione dell'applicazione**
   Il client e il server devono essere mandati in esecuzione. In questa fase si verifica il funzionamento dell'applicazione e il rispetto delle specifiche;

3. **Analisi del codice sorgente**
   Può essere richiesto di spiegare parti del codice e apportarvi semplici modifiche.

FUNZIONI DI UTILITÀ

Una documentazione di alto livello per le funzioni necessarie per leggere e scrivere file a blocchi possono essere visualizzate al seguente indirizzo:

https://www.programiz.com/c-programming/c-file-input-output