Informatica Generale II

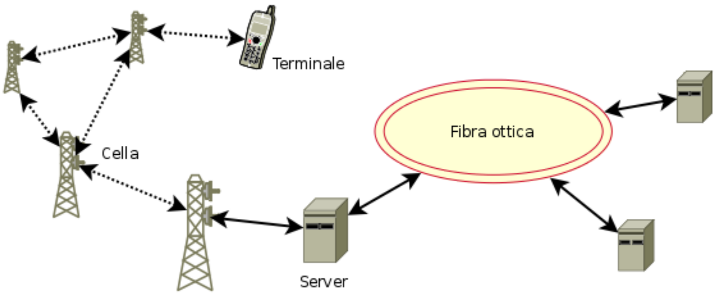
A.A. 2005/2006

Esame: 26 aprile 2007

Un server SMS

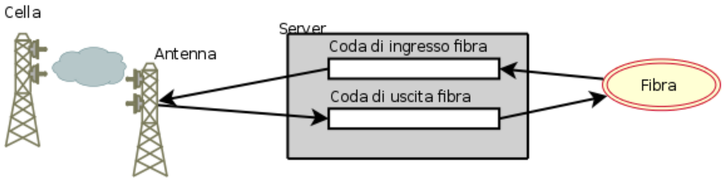
Gli `SMS` (Short Message Service) sono brevi messaggi di testo che possono essere inviati e ricevuti attraverso la rete di telefonia mobile. La rete e' composta di `celle` comunicanti tra loro, che coprono determinate aree geografiche del territorio. Ciascuna area e' gestita da un `server` che instrada il traffico tra la rete di celle e la `fibra ottica` che collega tutti i server della rete[1].

Quando un utente invia un messaggio SMS mediante il proprio telefono cellulare, il messaggio viene trasmesso alla cella piu' vicina, e da questa alla rete di celle, in un percorso che porta fino al server dell'area.



Il server riceve il messaggio da un'antenna e lo mette in una `coda di messaggi` di uscita alla fibra. Il messaggio resta nella coda fino a quando non viene processato, rimosso e spedito sulla fibra.

Quando un messaggio arriva dalla fibra, il messaggio viene messo nella coda di messaggi di ingresso dalla fibra. In seguito i messaggi verranno prelevati dalla coda e spediti alla cella attraverso l'antenna.



Il problema

Si vuole realizzare l'implementazione della parte del server che:

- Riceve i messaggi dall'antenna e lo inserisce nella coda di uscita su fibra.
- Riceve i messaggi dalla fibra, e inserisce il messaggio nella corrispondente coda di ingresso da fibra.
- Estrae un messaggio dalla coda di uscita su fibra e lo spedisce alla fibra.

---

[1] La descrizione qui riportata e' una semplificazione che non rispetta il modo in cui la trasmissione di SMS ad esempio sulla rete GSM e' implementata realmente.

- Estrae un messaggio dalla coda di ingresso da fibra e lo spedisce all'antenna.

Per l'implementazione si dovranno dichiarare le strutture dati necessarie, ed utilizzare il codice sorgente fornito. Il programma di test fornito definisce due server collegati da fibra ottica. Il primo server riceve in ingresso dall'antenna tre messaggi che trasferisce via fibra al secondo server. Il secondo server li spedisce poi ai destinatari attraverso la propria antenna.

Quesiti

1. **(Punti: 5)** Si implementino le definizione delle seguenti strutture dati. Nomi e tipi dei campi devono essere quelli richiesti.

   Messaggio costituito dei campi *testo* stringa di massimo 100 caratteri; *dest* intero senza segno che rappresenta il numero di telefono del destinatario del messaggio; *mitt* intero senza segno che rappresenta il numero di telefono del mittente del messaggio. Si implementi almeno un costruttore non di default.

   ListaMsg Lista singolarmente concatenata di messaggi. I campi sono *msg* e *next*. Si implementi almeno un costruttore non di default.

   CodaMsg Coda di messaggi FIFO basata su lista concatenata. Campi *lista* (contenente la lista di messaggi) e campo *len* di tipo intero, contenente la lunghezza della lista. Definire un costruttore di default.

   Server Un singolo server. Contiene i campi *nome* stringa di caratteri di tipo *const char**; *da_fibra* coda di messaggi in arrivo dalla fibra, diretti all'antenna; *a_fibra* coda di messaggi in arrivo dall'antenna, diretti alla fibra. Si implementi un costruttore che riceve il nome del server.

2. **(Punti: 4)** Si completino le funzioni: *Messaggio_stampa*, *ListaMsg_stampa*, *CodaMsg_stampa* e *Server_stampa*. Il formato richiesto e' indicato nel codice sorgente. Il parametro opzionale *Messaggio_stampa::stampa_testo* se true stampa anche il testo, altrimenti stampa solamente mittente e destinatario.

3. **(Punti: 7)** Si implementino le funzioni *CodaMsg_push* e *CodaMsg_pop* che facciano rispettivamente l'inserimento e la rimozione di un messaggio dalla coda FIFO. Il pushing si faccia in testa alla lista, il popping dal fondo della lista. Oltre ad effettuare l'inserimento e la rimozione del messaggio, le funzioni devono anche incrementare e decrementare il campo *CodaMsg::len*.

4. **(Punti: 4)** Si implementino le funzioni:

   - *Server_ricevi_da_antenna* che ricevuto un messaggio da un server, ne faccia il push sulla coda *a_fibra*
   - *Server_invia_ad_antenna* Che dato un server faccia:
     (a) Il pop di un messaggio dalla coda *da_fibra*
     (b) Stampi il messaggio (senza il testo) con il formato:
     `<nome server>: Inviato ad antenna messaggio: <messaggio>`
   - *Server_ricevi_da_fibra* che dato un server e un messaggio faccia il pushing del messaggio nella coda *da_fibra* del server.
   - *Server_invia_a_fibra* che dato un server:
     (a) Faccia il popping di un messaggio dalla coda *a_fibra*
     (b) Stampi il messaggio (senza il testo) nel formato:
     `<nome server>: Inviato a fibra messaggio: <messaggio>`
     (c) Restituisca il Messaggio tolto dalla coda.

Note importanti

1. Nella directory di lavoro *sim*/`Esame` vi è il file `main.cpp` . che il candidato dovrà completare nelle sue parti mancanti e con i dati personali.
2. Al fine della valutazione verrà preso in considerazione unicamente il file *sim*/`Esame/ main.cpp` . Ogni altro file verrà ignorato. Il candidato dovrà perciò assicurarsi di modificare e salvare unicamente tale file.

```
/*
  ----------------------------------------------------------------------
  NOME:
  CONGNOME:
  MATRICOLA:
  CODICE CALCOLATORE:
  ----------------------------------------------------------------------
*/

#include <iostream>
#include <cstdlib>
#include <assert.h>

using namespace std;

/* --------------------------- INIZIO PUNTO 1  ------------------------- */
/*                    Scrivere qui sotto la soluzione                    */
/*    vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv   */


// Singolo messaggio
struct Messaggio {
        //...
};


// Lista di messaggi

// Coda di messaggi

// Server
struct Server {
        const char* nome;
        //...
};



/*    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^   */
/* ---------------------------- FINE PUNTO 1  -------------------------- */



/* --------------------------- INIZIO PUNTO 2  ------------------------- */
/*                    Scrivere qui sotto la soluzione                    */
/*    vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv   */

/** Stampa il messaggio passato.
                se stampa_testo e’ false, non stampa il testo del messaggio.
                Formato richiesto:
Da <mitt> per <dest>: <testo>\n

    Se stampa_testo e’ true, altimenti:
Da <mitt> per <dest>\n

   Ad esempio:
Da 123 a 321: Ciao!

   Notare il newline alla fine della riga.
 */
void Messaggio_stampa(const Messaggio &msg, bool stampa_testo=true)
{
        // ...
}


/** Stampa la lista di messaggi. Ogni messaggio su una riga diversa.
                Ad esempio:
Da 123 a 321: Ciao!
Da 321 a 123: Com’e’ andato l’esame?
Da 123 a 321: Bene grazie!
 */
void ListaMsg_stampa(/* ... */)
{
}

/** Stampa la coda di messaggi, in qualsiasi ordine si desideri.
    Il formato e’:
```

```
Ci sono <len> messaggi in coda:
<msg1>
<msg2>
...
<msgN>

   Se stampa_messagi e’ true e la coda non e’ vuota, altrimenti:
Ci sono <len> messaggi in coda

   Si noti il newline alla fine.
*/
void CodaMsg_stampa(const CodaMsg &coda, bool stampa_messaggi)
{
        // ...
}

/** Stampa informazioni sul server.
                Il formato e’:
Server <nome>:
Da fibra: <coda da_fibra>
A fibra: <coda a_fibra>

   Il parametro stampa_messaggi deve essere passato a CodaMsg_stampa.
*/
void Server_stampa(/* ... ,*/ bool stampa_messaggi)
{
        // ...
}


/*    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^   */
/* ---------------------------- FINE PUNTO 2 -------------------------- */



/* --------------------------- INIZIO PUNTO 3  ------------------------- */
/*                    Scrivere qui sotto la soluzione                    */
/*    vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv   */
void CodaMsg_push(CodaMsg &coda, const Messaggio &msg)
{
        // ...
}


/* ... */ CodaMsg_pop(/*... */)
{

}


/*    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^   */
/* ---------------------------- FINE PUNTO 3 -------------------------- */



/* --------------------------- INIZIO PUNTO 4  ------------------------- */
/*                    Scrivere qui sotto la soluzione                    */
/*    vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv   */


/** Riceve dall’antenna un messaggio */
void Server_ricevi_da_antenna(Server &server, const Messaggio &msg)
{
        // ...
}


/** Rimuove un messaggio dalla coda da_fibra, e lo invia
                all’antenna. Stampa un testo */
void Server_invia_ad_antenna(/* ... */)
{
        // ...
}

/** Riceve un messagio dalla fibra */
void Server_ricevi_da_fibra(/* ... */)
{
}
```

```
/** Rimuove un messaggio dalla coda a_fibra, e restituisce il
                messaggio inviato. Stampa un testo. */
/* ... */ Server_invia_a_fibra(/* ... */)
{
        // ...
}



/*     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^   */
/* ---------------------------- FINE PUNTO 4 -------------------------- */



// ----------------------------------------------------------------------
// Main del programma
// ----------------------------------------------------------------------

int main()
{
        // i due server
        Server s1("Trento");
        Server s2("Verona");

        // 3 messaggi
        Messaggio messaggi[] = {
                Messaggio("Taca la pasta che ’rivo", 123, 321),
                Messaggio("Vaben, te ’speto!", 321, 123),
                Messaggio("Ciao", 321, 213)
        };

        // 3 cellulari mandano un messaggio
        for (int i=0; i<(sizeof(messaggi)/sizeof(messaggi[0])); ++i) {
                Server_ricevi_da_antenna(s1, messaggi[i]);
        }


        // Stampa server s1 e s2:
        Server_stampa(s1, true);
        cout << endl;
        Server_stampa(s2, true);

        // invia i messaggi sulla fibra, e trasferisci sul server s2
        for (int i=0; i<(sizeof(messaggi)/sizeof(messaggi[0])); ++i) {
                Messaggio msg = Server_invia_a_fibra(s1);
                Server_ricevi_da_fibra(s2, msg); // trasferisci a s2
        }

        // Stampa server s1 e s2:
        cout << endl;
        Server_stampa(s1, true);
        cout << endl;
        Server_stampa(s2, true);


        // trasferisci da s2 all’antenna
        for (int i=0; i<(sizeof(messaggi)/sizeof(messaggi[0])); ++i) {
                Server_invia_ad_antenna(s2);
        }


        // Stampa server s1 e s2:
        cout << endl;
        Server_stampa(s1, true);
        cout << endl;
        Server_stampa(s2, true);

  return 0;
}
```