```c
#include "Client.h"

int sockfd,ingame=0,logged=0, connected=0;
int main(int argc, char* argv[]){
  signal(SIGINT, handleSignal);
  clear();
  int n_b_r;
  char msg[500],input[100],c;
  if(argc!=3){
    printf("Passa indirizzo e porta\n");
    return(0);
  }
  struct sockaddr_in server_addr;
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(atoi(argv[2]));
  inet_aton(argv[1],&server_addr.sin_addr);
  if ((sockfd=socket(AF_INET,SOCK_STREAM,0))<0) {
    printf("Errore apertura socket");
  }else{
    if(connect(sockfd,(struct sockaddr *)&server_addr,
sizeof(server_addr))<0){
      printf("Errore connessione socket %d\n", errno);
    }else{
      connected=1;
      logged=comunication(sockfd);
      if (logged<0) {
        clear();
        if(logged!=-5)
          printf("Al momento risulta impossibile eseguire
operazione di Log-In e Sign-In.\nRiprova tra poco.\n");
      }else{
        clear();
        ingame=comunicationGame(sockfd);
      }
    }
  }
  printf("Arrivederci.\n");
  close(sockfd);
  return 0;
}

void handleSignal(int sig){
  if(sig==SIGINT){
    clear();
    fflush(STDIN_FILENO);
    if(logged==1)
```

```
45        write(sockfd,USER_LOG_OUT,strlen(USER_LOG_OUT));
46      else if(connected==1)
47        write(sockfd,CLIENT_GONE,strlen(CLIENT_GONE));
48      close(sockfd);
49      system("reset");
50      printf("Arrivederci.\n");
51      exit(-1);
52    }
53  }
54
55  int comunication(int sockfd){
56    int n_b_r,c;
57    char msg[250],input[1000];
58    while(1){
59      leggi();
60      if(goOn(msg)){
61        return 1;
62      }else{
63        if( (strcmp(msg, "-1")==0) ){
64          return -1;
65        }else{
66          if (strcmp(msg,WELCOME_MESSAGE)==0) {
67            scanf("%1s",input);
68            if ( (strcmp(input,"e")==0) || (strcmp(input,"E"
    )==0) ) {
69              raise(SIGINT);
70            }
71          }else{
72            scanf("%20s",input);
73          }
74          while((c = getchar()) != '\n' && c != EOF);
75          write(sockfd,input,strlen(input));
76          clear();
77        }
78      }
79    }
80  }
81
82  int comunicationGame(int sockfd){
83    int n_b_r;
84    struct termios orig;
85    clear();
86    char msg[4000],input[100];
87    struct termios raw;
88    tcgetattr(STDIN_FILENO, &raw);
89    orig = raw;
```

```
 90   raw.c_lflag &= ~(ECHO | ICANON);
 91   tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);
 92   while(1){
 93     leggi();
 94     if(strcmp(msg,"GETOUT")==0){
 95       break;
 96     }else if((strcmp(msg, VICTORY_MESSAGE)==0) || (strcmp(
    msg, LOSS_MESSAGE)==0)){
 97       clear();
 98       printf("%s", msg);
 99       sleep(4);
100       break;
101     }
102     read(STDIN_FILENO, input, 1);
103     write(sockfd,input,1);
104     fflush(STDIN_FILENO);
105     clear();
106   }
107   clear();
108   tcsetattr(STDIN_FILENO, TCSAFLUSH, &orig);
109   return 0;
110 }
111
112 int goOn(char msg[]){
113   return ((strcmp(msg,SUCCESS_MESSAGE_LIM)==0) || (strcmp(
    msg,SUCCESS_MESSAGE_SIM)==0));
114 }
115
```

```c
#include <sys/socket.h>
#include <sys/un.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <signal.h>
#include <errno.h>
#include "termios.h"
#include "../costanti.h"
#include "../messaggi.h"

/*
  La funzione comunication si occupa di leggere i messaggi inviati dal Server durante il login e la registrazione.
  Inoltre scrive le risposte del client sulla socket. Nel caso l'utente decida di uscire, premendo il tasto 'e' lancia un segnale di SIGINT.
  Il controllo sull'effettivo login è svolto dalla funzione goOn.
  Parametri:
    int sockfd, il fd della socket;
  Valori di Ritorno:
    1, se l'utente si è loggato con successo;
    -1, altrimenti.
 */
int comunication(int sockfd);

/*
  La funzione communicationGame si occupa di gestire la comunicazione con il Server durante la partita. Imposta il terminale in modalita no ECHO e
  disabilita ICANON, così che i caratteri inseriti vengano letti senza dover premere il carattere di newline. Al termine della partita, cioè quando
  l'utente decide d'uscire O la partita è terminata reimposta il terminale alle impostazioni standard.
  Parametri:
    int sockfd, il fd della socket;
 */
int comunicationGame(int sockfd);
```

```
38
39 /*
40   La funzione di handleSignal gestisce l'evento di chiusura
    forzata del terminale. Nel caso l'utente sia loggato
   specifica al Server l'utente che si
41   è appena disconesso, altrimenti, se l'utente è connesso
   ma non autenticato invia un messaggio al Server per far
   partire la gestione dell'evento
42   'disconessione Client non autenticato'. In entrambi i
   casi chiude la socket di comunicazione e termina il
   processo.
43   Parametri:
44     int sig, il segnale da gestire.
45  */
46 void handleSignal(int sig);
47
48 /*
49   La funzione goOn riceve un messaggio come parametro e
   controlla se corrisponde ad un login o ad una registrazione
    avvenuta con successo.
50   Parametri:
51     char[], il messsaggio da controllare;
52   Valori di Ritorno:
53     1, se il login o la registrazione sono andati a buon
   fine;
54     0, altrimenti;
55  */
56 int goOn(char []);
57
```