```
/*
 * Corso di Fondamenti di Informatica
 * Esercizio 1bis:
 * riscrittura corretta, ovverosia
 *   1. chiara
 *   2. con nomi di variabile esplicativi
 *   3. con l'uso di parametri (#define)
 *   4. commentata
 * del programma dell'esercizio 1
 */

/* stampa a schermo il massimo voto di laurea ottenibile da
 * uno studente in base al suo voto medio negli esami, al numero di
 * lodi conseguite e al fatto che presenti una tesi o una tesina */

/* Fonte per l'algoritmo di calcolo del voto di laurea:
   http://www.ingindinf.polimi.it/fileadmin/files/pdf_scuola/regolamenti_lauree/RegolamentoProvaFinale.pdf
 */


#include <stdio.h>

/*****************************************************************
 * NOTA:
 * per il corretto funzionamento del programma e' necessario che tutti i
 * parametri sotto definiti siano espressi secondo la notazione dei valori
 * di tipo float del C (ovvero esplicitando il punto e la parte decimale).
 * In particolare, nel caso di valori interi va esplicitato il .0 in coda.
 *****************************************************************/

#define MAX_INCREMENTO_TESI 7.0
/* punteggio massimo assegnabile all'esame di laurea nel caso di tesi */
#define MAX_INCREMENTO_TESINA 4.0
/* punteggio massimo assegnabile all'esame di laurea nel caso di tesina
 * (ovvero tesi senza controrelatore) */
#define MAX_VOTO_ESAME 30.0
/* voto massimo per il singolo esame, esclusa la eventuale lode */
#define MAX_VOTO_LAUREA 110.0
/* voto massimo di laurea */
#define SOGLIA_SENZA_LODI 113.0
/* punteggio minimo che lo studente deve conseguire all'esame di laurea per
```

```
 * poter conseguire la lode nel caso in cui non abbia conseguito alcuna lode
 * negli esami */
#define BONUS_LODE 0.5
/* quando lo studente ha conseguito lodi negli esami, la soglia per
 * ottenere la lode viene abbassata di BONUS_LODE per ciascuna lode
 * conseguita */
#define SOGLIA_MINIMA_LODE 111.0
/* la soglia per ottenere la lode non può comunque mai essere
 * inferiore a SOGLIA_MINIMA_LODE */

/*************************************************************************
 * NOTA IMPORTANTE
 * Quelli soprastanti sono due esempi di DEFINIZIONE DI PARAMETRI
 * in un programma C. Un PARAMETRO è un valore che determina le
 * modalità di funzionamento del programma, e che può cambiare se
 * cambiano le condizioni che descrivono il problema che il
 * programma risolvere. Cambiare i valori dei parametri consente di
 * adattare il programma ad una variazione delle caratteristiche
 * del problema che e' stato progettato per risolvere, ma non puo'
 * adattare il programma ad un problema di tipo differente. I
 * parametri sono spesso numerici, ma non necessariamente: ad
 * esempio un parametro può avere la forma di un carattere o di una
 * stringa.
 * OGNI VALORE PRESENTE IN UN PROGRAMMA CHE POSSA, ANCHE SOLO
 * CONCEPIBILMENTE, SUBIRE UNA MODIFICA SENZA CHE IL PROBLEMA
 * CONSIDERATO CAMBI NATURA *DEVE* ESSERE INTRODOTTO COME
 * PARAMETRO!!!
 *
 * Ad esempio il numero di elementi di un insieme di dati
 * (ad es. la dimensione di un array) DEVE essere introdotto come
 * parametro, perché l'utente del programma potrebbe volerlo
 * modificare. Un simile parametro potrebbe chiamarsi ad esempio
 * NUM_ELEM_ARRAY. Invece è accettabile che il numero di giorni
 * della settimana, ovvero 7, sia inserito nel programma
 * direttamente in forma numerica, dato che non è immaginabile che
 * possa essere modificato; sebbene sia opportuno l'uso di un
 * parametro perfino in questo caso. Analogamente è accettabile
 * usare in un programma il valore numerico (0) dell'indice del
 * primo elemento di un array, perché tale valore è immutabile per
 * la natura stessa del linguaggio C; mentre NON VA BENE inserire
 * in forma numerica il valore dell'indice dell'ultimo elemento.
```

```
 * Al posto di questo valore numerico va usata un'espressione
 * basata sul parametro che dà la dimensione dell'array:
 * nell'esempio precedente questa espressione sarà
 * (NUM_ELEMENTI_ARRAY-1).
 ************************************************************************/

int main()
{
  float MediaEsami;
  /* media esami, espressa in termini di punteggio d'esame (ad es.
   * trentesimi) */
  int NumeroLodi;
  /* numero di lodi conseguite dallo studente negli esami */
  char TesiOTesina;
  /* tipo di laurea: 't' se esame con tesina, 'T' se con tesi */
  float MediaPerLaurea;
  /* media esami, espressa in termini di punteggio di laurea (ad es.
   * centesimi), non arrotondata o troncata */
  float PunteggioLaurea;
  /* punteggio utilizzato per calcolare il voto di laurea */
  float SogliaLode;
  /* soglia che PunteggioLaurea deve superare perche' allo
   * studente possa essere assegnata la lode */
  int MaxVotoOttenibile;
  /* massimo voto di laurea raggiungibile dallo studente, da
   * calcolare; per definizione si tratta di un numero intero */
  char PossibileLode;
  /* 'L' se e' possibile ottenere la lode, ' ' altrimenti */

  printf("Inserisci in successione:\nla media dei voti d'esame (lodi escluse);\nil numero di lodi conseguite negli
esami;\nSe presenti una tesi (T) o una tesina (t).\n");

  scanf("%f", &MediaEsami);
  scanf("%d", &NumeroLodi);
  scanf("%*c%c", &TesiOTesina);
  /* NOTA: nella istruzione scanf qui sopra, la presenza
   * dell'espressione di controllo "*c" fa in modo che scanf, prima
   * di effettuare la lettura di carattere indicata dall'espressione
   * di controllo "%c", legga un carattere e lo elimini.
   * Questo meccanismo e' utilizzato per eliminare il \n dovuto alla
   * scanf precedente, che altrimenti - in quanto char - sarebbe
```

```
 * stato letto e immagazzinato nella variabile TesiOTesina al
 * posto del carattere inserito via tastiera dall'utente del
 * programma.
 * Questo meccanismo e' utile quando un programma usa scanf per
 * eseguire piu' di una lettura da tastiera, ed alcuni dei dati
 * letti sono di tipo char. */

MediaPerLaurea = MediaEsami * (MAX_VOTO_LAUREA / MAX_VOTO_ESAME);
/* conversione della media esami nel formato dei voti di laurea;
 * notare che (essendo MediaEsami e MAX_VOTO_LAUREA dei float)
 * il risultato della divisione è un float e dunque, per eseguire
 * la moltiplicazione, il C converte in float anche
 * MAX_VOTO_LAUREA e l'espressione produce un risultato di tipo
 * float */

if ('t' == TesiOTesina)
{
  PunteggioLaurea = MediaPerLaurea + MAX_INCREMENTO_TESINA;
}
else
{
  PunteggioLaurea = MediaPerLaurea + MAX_INCREMENTO_TESI;
}

SogliaLode = SOGLIA_SENZA_LODI - BONUS_LODE * NumeroLodi;

if (SogliaLode < SOGLIA_MINIMA_LODE )
{
  SogliaLode = SOGLIA_MINIMA_LODE;
}

if (PunteggioLaurea >= SogliaLode)
{
  PossibileLode = 'L';
}
else
{
  PossibileLode = ' ';
}

/* occorre ora arrotondare PunteggioLaurea all'intero piu'
```

```
   * vicino: a tale scopo sfruttiamo il meccanismo di conversione
   * automatica di tipo del C per troncare la parte frazionaria,
   * ovverosia la parte dopo la virgola, della media */

  MaxVotoOttenibile = PunteggioLaurea;
  /* dal momento che MaxVotoOttenibile e' un dato di tipo int,
   * questo assegnamento elimina la parte dopo la virgola di
   * PunteggioLaurea (viene cioe' eseguito un troncamento) */

  if (PunteggioLaurea - MaxVotoOttenibile >= 0.5)
  /* (PunteggioLaurea - MaxVotoOttenibile) e' la parte di
   * PunteggioLaurea che e' stata troncata: se e' minore di 0.5
   * arrotondamento e troncamento producono lo stesso valore,
   * altrimenti no. Nel secondo caso il valore arrotondato si
   * ottiene da quello troncato sommandogli 1. */
  {
    MaxVotoOttenibile = MaxVotoOttenibile + 1;
  }

  /* Nota: un approccio diverso, ma ugualmente valido, al problema di
   * arrotondare il valore di PunteggioLaurea all'intero piu' vicino sarebbe
   * stato quello di sommare 0.5 a PunteggioLaurea e poi applicare il
   * troncamento al risultato della somma. Si lascia allo studente, per
   * esercizio, l'implementazione di questo metodo alternativo, da accompagnare
   * (naturalmente) con un commento che spieghi la ragione per cui le
   * operazioni eseguite -di per se' poco esplicative- vengono compiute */

  /* infine, occorre assicurarsi che MaxVotoOttenibile non superi
   * il valore massimo ammissibile: */
  if (MaxVotoOttenibile > MAX_VOTO_LAUREA)
  {
    MaxVotoOttenibile = MAX_VOTO_LAUREA;
  }

  printf("\nMassimo voto di laurea raggiungibile dallo studente: %d%c", MaxVotoOttenibile, PossibileLode);
  printf("\n\n");

  return(0);
}
```