```
/*
 * Corso di Fondamenti di Informatica
 * Esercizio:
 * tema d'esame (semplificato)
 */

/*
   Si scriva un programma che
   - legge da tastiera una stringa di al piÃ¹ 100 caratteri priva di spazi;
   - genera una lista dinamica di caratteri contenente i caratteri inseriti dall'utente;
   - cancella lâ€™elemento della lista situato in posizione centrale.
   Nella ricerca dell'elemento da eliminare, il programma deve scandire la lista stessa solo una volta.
   Detto N il numero di elementi della lista, l'elemento in posizione centrale Ã¨ il K-esimo, dove K Ã¨ la parte intera di
(N+1)/2.
*/

/*****************************************************************************
 VERSIONE 1 DEL PROGRAMMA
 Viene utilizzato un apposito puntatore (pPrecedeDaElim) per puntare
 all'elemento che precede quello da eliminare. Questo richiede un trattamento
 particolare del caso in cui l'elemento da eliminare e' il primo (dunque non
 esiste un elemento che lo precede).
*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#define MAX_CARATTERI 100
/* numero massimo di caratteri che l'utente puo' chiedere di inserire nella
   lista */

typedef struct EL
{
  char        Dato;
  struct EL*  pProssimo;

} ElemLista;
/* tipo di dato degli elementi della lista */


/* NOTA: le tre funzioni seguenti hanno implementazione identica nelle tre
   versioni di questo esercizio */
```

```
ElemLista* CreaLista(char* ElencoCaratteri);
/*** NON RICHIESTA ALL'ESAME ***/
/* crea in memoria dinamica una lista contenente i caratteri dell'array
 * ElencoCaratteri. L'ultimo elemento della lista Ã¨ quello che contiene il
 * carattere che precede il '\0' che ElencoCaratteri deve contenere.
 * Restituisce un puntatore al primo elemento della lista. */

void StampaElementi(ElemLista* pTesta);
/*** NON RICHIESTA ALL'ESAME ***/
/* riceve la testa di una lista e ne stampa il campo Dato degli elementi */

void StampaRiferimenti(ElemLista* pTesta);
/*** NON RICHIESTA ALL'ESAME ***/
/* stampa riferimenti utili per identificare l'elemento centrale */


int main()
{
  char ContenutoLista[MAX_CARATTERI+1];
  /* contiene i caratteri da inserire nella lista seguiti da '\0' */
  ElemLista* pTesta;
  /* testa della lista */
  ElemLista* pCorrente;
  /* puntatore all'elemento corrente della lista */
  ElemLista* pPrecedeDaElim;
  /* punta all'elemento che precede quello da eliminare; vale NULL se l'elemento
   * da eliminare Ã¨ il primo della lista */
  ElemLista* pDaEliminare;
  /* punta all'elemento della lista dinamica che va eliminato */
  int ContaPassi;
  /* utilizzato per far avanzare pPrecedeDaElim ogni 2 avanzamenti di
   * pCorrente; assume valori alternati +1 e -1 */

  printf("\nInserisci una stringa composta da non piu' di %d caratteri (no spazi) e premi <enter>\n", MAX_CARATTERI);
  scanf("%s", ContenutoLista);
  pTesta = CreaLista(ContenutoLista);

  printf("\nLista iniziale:\n");
  StampaElementi(pTesta);
  printf("\n");
```

```
StampaRiferimenti(pTesta);

/* L'algoritmo sfrutta il fatto che per individuare l'elemento centrale da
 * eliminare e' sufficiente esaminare gli elementi della lista con pCorrente
 * fino a giungere all'ultimo, e far avanzare pPrecedeDaElim a velocita'
 * dimezzata (ovvero di 1 elemento ogni volta che pCorrente ne percorre 2).
 * Nel momento in cui pCorrente raggiunge l'ultimo elemento, e' possibile
 * procedere all'eliminazione dell'elemento che segue quello puntato da
 * pPrecedeDaElim */

if (NULL != pTesta)
/* la lista non e' vuota */
{
  pCorrente = pTesta->pProssimo;
  /* punta al secondo elemento della lista, o vale NULL se la lista ha un
     unico elemento */
  ContaPassi = 1;
  pPrecedeDaElim = NULL;

  while (NULL != pCorrente)
  {
    /* frammento di codice utilizzato per debugging: inizio...
    printf("\n#DEBUG elemento corrente '%c'; individuato per eliminazione ", pCorrente->Dato);
    if (NULL == pPrecedeDaElim)
    {
      printf("'%c'", pTesta->Dato);
    }
    else
    {
      printf("'%c'", pPrecedeDaElim->pProssimo->Dato);
    }
    ...e fine */

    if (-1 == ContaPassi)
    {
      if (NULL == pPrecedeDaElim)
      {
        pPrecedeDaElim = pTesta;
      }
      else
      {
```

```
        pPrecedeDaElim = pPrecedeDaElim->pProssimo;
      }
    }

    ContaPassi = ContaPassi * -1;
    pCorrente = pCorrente->pProssimo;
  }

  /* eliminazione dell'elemento centrale della lista */
  if (NULL == pPrecedeDaElim)
  /* l'elemento da eliminare e' il primo della lista*/
  {
    pDaEliminare = pTesta;
    pTesta = pDaEliminare->pProssimo;
    free(pDaEliminare);
  }
  else
  {
    pDaEliminare = pPrecedeDaElim->pProssimo;
    pPrecedeDaElim->pProssimo = pDaEliminare->pProssimo;
    free(pDaEliminare);
  }
}

printf("\nLista privata dell'elemento centrale:\n");
StampaElementi(pTesta);
printf("\n\n");

return(0);
}


ElemLista* CreaLista(char* ElencoCaratteri)
{
int Cont;
/* usato per scorrere ElencoCaratteri */
ElemLista* pTestaLista;
/* testa della lista, da restituire al programma chiamante */
ElemLista* pElementoCorrente;
/* punta all'ultimo elemento aggiunto alla lista */
```

```
  pTestaLista = NULL;
  Cont = 0;

  while ( '\0' != ElencoCaratteri[Cont] )
  {
    if ( NULL == pTestaLista )   /* la lista Ã¨ vuota */
          {
            pTestaLista = malloc(sizeof(ElemLista));
      pElementoCorrente = pTestaLista;
      /* ora pElementoCorrente punta all'elemento appena creato */
      pElementoCorrente->Dato = ElencoCaratteri[Cont];
      pElementoCorrente->pProssimo = NULL;
          }
    else   /* la lista non Ã¨ vuota */
    {
      pElementoCorrente->pProssimo = malloc(sizeof(ElemLista));
      pElementoCorrente = pElementoCorrente->pProssimo;
      /* ora pElementoCorrente punta all'elemento appena creato */
      pElementoCorrente->Dato = ElencoCaratteri[Cont];
      pElementoCorrente->pProssimo = NULL;
    }

    ++Cont;
  }

  return(pTestaLista);
}


void StampaElementi(ElemLista* pTesta)
{
  if ( NULL != pTesta )
  {
    printf("%c; ", pTesta->Dato);

    StampaElementi(pTesta->pProssimo);
    /* chiamata ricorsiva alla funzione StampaElementi */
  }
}
```

```
void StampaRiferimenti(ElemLista* pTesta)
{
  int Cont;
  /* contatore usato per contare i caratteri contenuti nella lista */
  int NumCaratteri;
  /* numero totale di caratteri contenuti nella lista */
  ElemLista* pCorrente;
  /* puntatore all'elemento corrente della lista */

  Cont = 1;
  for (pCorrente = pTesta; (NULL != pCorrente); pCorrente = pCorrente->pProssimo)
  {
    printf("%d  ", Cont%10);
    /* in questo modo viene stampato un numero che quando raggiunge 10 torna
       a 0, restando pertanto composto da una sola cifra decimale */
    ++Cont;
  }

  NumCaratteri = Cont-1;

  printf("\n");
  for(Cont = 1; Cont < (NumCaratteri+1)/2; ++Cont)
  {
    printf("   ");
  }
  printf("^");
}


/*****************************************************************************
 Prova di esecuzione:

  Inserisci una stringa composta da non piu' di 100 caratteri (no spazi) e premi <enter>
  UnaListaDiProva!!!

  Lista iniziale:
  U; n; a; L; i; s; t; a; D; i; P; r; o; v; a; !; !; !;
  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8
                         ^
  Lista privata dell'elemento centrale:
  U; n; a; L; i; s; t; a; i; P; r; o; v; a; !; !; !;
```

/**********