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

/*
Si scriva una procedura che, ricevendo in ingresso una lista dinamica di caratteri, cancelli lâ€™elemento in posizione
centrale, scandendo 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 3 DEL PROGRAMMA
 Rispetto alla versione 2, in cui l'uso di puntatori a puntatore era una scelta,
 nella versione 3 esso e' indispensabile. L'esercizio d'esame richiede infatti
 di scrivere una procedura: tale procedura deve dunque essere in grado di
 eliminare l'elemento centrale della lista in qualsiasi caso, incluso quello in
 cui esso e' il primo elemento della lista. A tal fine e' necessario che la
 procedura sia in grado di modificare la testa della lista, che percio' deve
 ricevere come parametro d'ingresso passato per indirizzo. Dato che la testa
 e' di tipo ElemLista*, il parametro sarÃ  dunque di tipo ElemLista**.
*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#define MAX_CARATTERI 100
/* dato che l'esercizio non richiede di scrivere il codice che produce la lista,
   utilizzeremo una funzione che legge da tastiera una stringa e crea la lista
   contenente i caratteri della stringa */

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

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


void EliminaCentrale(ElemLista** ppTesta);
```

```
/* riceve un puntatore alla testa di una lista dinamica ed elimina l'elemento
 * centrale della lista, ovverosia quello in posizione (N+1)%2 dove N e' il
 * numero di 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 */

  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);

  EliminaCentrale(&pTesta);
```

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

  return(0);
}


void EliminaCentrale(ElemLista** ppTesta)
/* 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 ppPuntaADaElim a velocita'
 * dimezzata (ovvero di 1 elemento ogni volta che Corrente ne percorre 2).
 * Nel momento in cui Corrente raggiunge l'ultimo elemento, e' possibile
 * procedere all'eliminazione dell'elemento puntato da DaEliminare */

/* NOTA: il codice che segue corrisponde esattamente a quello della versione 2
   del programma a meno della sostituzione di ppTesta al posto di &pTesta */
{
  ElemLista* pCorrente;
  /* punta all'elemento correntemente in esame della lista */
  ElemLista** ppPuntaADaElim;
  /* punta al puntatore che punta all'elemento da eliminare */
  ElemLista* pDaEliminare;
  /* punta all'elemento della lista dinamica che va eliminato */
  int ContaPassi;
  /* utilizzato per far avanzare ppPuntaADaElim ogni 2 avanzamenti di pCorrente;
     assume valori alternati +1 e -1 */

  if (NULL == *ppTesta)
  /* la lista e' vuota, non occorre fare nulla */
  {
    return;
  }

  pCorrente = (*ppTesta)->pProssimo;
  /* punta al secondo elemento della lista, o vale NULL se la lista ha un
     unico elemento */
  ContaPassi = 1;
  ppPuntaADaElim = ppTesta;
```

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

    if (-1 == ContaPassi)
    {
      ppPuntaADaElim = &((*ppPuntaADaElim)->pProssimo);
      /* nota: le parentesi intorno a *ppPuntaADaElim sono necessarie per
         evitare un errore in compilazione: infatti l'operatore -> ha
         precedenza sull'operatore * */
    }

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

  /* eliminazione dell'elemento centrale della lista */
  pDaEliminare = *ppPuntaADaElim;
  *ppPuntaADaElim = pDaEliminare->pProssimo;
  free(pDaEliminare);
}


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("^");
}
```