```
#include <stdio.h>
#define CAPIENZA 4
#define TERMINATORE '-'

typedef struct
{
  char Dati[CAPIENZA];
  int Utili;
} TipoL;
/* tipo lista sequenziale utile per contenere una sequenza di char */

typedef enum {false, true} boolean;

void Stampa(TipoL ListaDaStampare);
/* riceve una lista sequenziale di char e ne stampa il contenuto */

char Minimo(TipoL ListaDaEsaminare, char Soglia);
/* riceve una lista sequenziale di char e restituisce il minimo tra i valori maggiori di Soglia in
essa contenuti. Se la lista non contiene valori superiori a Soglia, la funzione restituisce
TERMINATORE. */

void EseguiInput(TipoL *ListaDaRiempire);
/* riceve una lista sequenziale di char e inserisce in essa i dati richiedendoli all'utente */

void Ordina(TipoL *ListaDaOrdinare);
/* riceve una lista sequenziale di char contenente caratteri minuscoli tutti diversi tra loro e ne
ordina i dati per valori crescenti. La funzione opera correttamente solo se la lista contiene almeno
un carattere. */

int main()
{
  TipoL Lista; /* lista sequenziale di interi inseriti dall'utente */

  /*** FASE DI INPUT ***/
  EseguiInput(&Lista);

  /*** FASE DI STAMPA ***/
  Stampa(Lista);

  /*** FASE DI STAMPA DEL MINIMO ***/
  if (Lista.Utili > 0)
  {
    printf("\nMinimo: %c", Minimo(Lista, 'a'-1));
  }
  /* nota: dato che i dati nella lista sono caratteri minuscoli, essi sono tutti maggiori o uguali ad
'a': dunque usare una soglia pari ad 'a'-1 trova il minimo tra tutti i dati nella lista */

  /*** FASE DI ORDINAMENTO***/
  Ordina(&Lista);

  /*** FASE DI STAMPA (DOPO L'ORDINAMENTO) ***/
  Stampa(Lista);


  return (0);
}

void Stampa(TipoL ListaDaStampare)
{
  int ContaStampati;
  /* numero di elementi della lista stampati a schermo */

  printf("\n\nContenuto della lista:\n{ ");
  for (ContaStampati = 0; ContaStampati < ListaDaStampare.Utili; ++ContaStampati)
```

```
  {
    printf("%c ", ListaDaStampare.Dati[ContaStampati]);
  }
  printf("}\n");
}

char Minimo(TipoL ListaDaEsaminare, char Soglia)
{
  char Min;
  /* elemento di minimo valore da restituire; vale TERMINATORE se la lista non contiene dati
superiori a Soglia */
  int ContaElem;
  /* usato per scorrere il campo Dati */

  Min = TERMINATORE;
  for (ContaElem = 0; ContaElem < ListaDaEsaminare.Utili; ++ContaElem)
  {
    if (ListaDaEsaminare.Dati[ContaElem] > Soglia)
    {
      if ((TERMINATORE == Min) || (ListaDaEsaminare.Dati[ContaElem] < Min))
      {
        Min = ListaDaEsaminare.Dati[ContaElem];
      }
    }
  }

  return (Min);
}

void EseguiInput(TipoL *ListaDaRiempire)
{
  boolean FineInput;
  /* controlla la fine della fase di input */
  int ContaElementi;
  /* conta gli elementi della lista gia' esaminati */
  boolean GiaPresente;
  /* vale true se nuovo elemento e' un duplicato di un elemento
   * gia' presente nella lista */

  ListaDaRiempire->Utili = 0;
  /* svuota la lista */

  printf("\n\nInserisci da 0 a %d caratteri minuscoli diversi tra loro.", CAPIENZA);
  printf("\n[Il carattere '%c' termina l'input. Ogni altro carattere sara' ignorato.]\n",
TERMINATORE);

  FineInput = false;

  while ((false == FineInput) && ListaDaRiempire->Utili < CAPIENZA)
  {
    scanf("%c", &ListaDaRiempire->Dati[ListaDaRiempire->Utili]);
    scanf("%*c");
    /* elimina il carattere '\n' inserito dall'utente premendo <enter> e che
     * altrimenti verrebbe letto alla prossima iterazione del ciclo */

    if ( (ListaDaRiempire->Dati[ListaDaRiempire->Utili] >= 'a') && (ListaDaRiempire-
>Dati[ListaDaRiempire->Utili] <= 'z') )
    /* se il valore inserito Ã¨ utile */
    {
      /* verifica se il valore inserito era giÃ  presente */
      ContaElementi = 0;
      GiaPresente = false;

      while (ContaElementi < ListaDaRiempire->Utili)
      {
```

```
        /* printf("\nDEBUG: sto esaminando l'elemento di indice %d \n", ContaElementi); */

        if (ListaDaRiempire->Dati[ContaElementi] == ListaDaRiempire->Dati[ListaDaRiempire->Utili])
        /* se l'elemento in esame e' identico a quello che si vuole
         * inserire nella lista sequenziale */
        {
          GiaPresente = true;
        }
        ++ContaElementi;
      }

      if (false == GiaPresente)
      {
        ++ListaDaRiempire->Utili;
        /* solo se ListaDaRiempire->Utili viene incrementato si completa
         * effettivamente l'inserimento del nuovo elemento */
      }
      else
      {
        printf("Il carattere '%c' e' un doppione ed e' stato ignorato.\n", ListaDaRiempire-
>Dati[ListaDaRiempire->Utili]);
      }
    }
    else
    {
      if ( TERMINATORE == ListaDaRiempire->Dati[ListaDaRiempire->Utili] )
      {
        FineInput = true;
      }
      else
      {
        printf("Il carattere '%c' non e' valido ed e' stato ignorato.\n", ListaDaRiempire-
>Dati[ListaDaRiempire->Utili]);
      }
    }
  }
}

void Ordina(TipoL *ListaDaOrdinare)
{
  TipoL ListaOrdinata;
  /* nuova lista i cui elementi sono in ordine crescente */
  int ElementoCorrente;
  /* indice dell'elemento di ListaOrdinata attualmente in elaborazione */

  ListaOrdinata.Dati[0] = Minimo(*ListaDaOrdinare, 'a'-1);
  /* inserisce il primo elemento nella lista ordinata */
  /* Nota: dato che i dati nella lista sono caratteri minuscoli, essi sono tutti maggiori o uguali ad
'a': dunque usare una soglia pari ad 'a'-1 trova il minimo tra tutti i dati nella lista */
  /* Nota: la lista contiene sempre almeno un elemento */

  ListaOrdinata.Utili = 1;

  for (ElementoCorrente = 1; ElementoCorrente < ListaDaOrdinare->Utili; ++ElementoCorrente)
  {
    ListaOrdinata.Dati[ElementoCorrente] = Minimo(*ListaDaOrdinare,
ListaOrdinata.Dati[ListaOrdinata.Utili-1]);
    /* il secondo membro Ã¨ il dato di ListaDaOrdinare avente valore minimo tra quelli maggiori
dell'ultimo dato inserito in ListaOrdinata; si noti
       che non esiste il rischio di saltare eventuali elementi uguali ad
       altri gia' inseriti nella lista ordinata perche' per costruzione
       la lista originale non contiene elementi uguali tra loro */

    ++ListaOrdinata.Utili;
  }
```

```
  *ListaDaOrdinare = ListaOrdinata;
}
```