Calcolatori Elettronici A.A. 2019/2020
Prova di laboratorio 19/12/2019 - Turno 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 * Scrivere le funzioni "sparseList" e "sparseRow" (ed eventuali funzioni
 * ausiliarie) in C.
 * La funzione sparseList riceve come parametri di ingresso una matrice M sparsa
 * (ovvero una matrice i cui valori sono quasi tutti uguali a 0) di interi,
 * il numero di righe di M ed il numero di colonne di M;
 * la funzione restituisce in output una lista L1 contenente, per ogni elemento
 * di M diverso da zero, il valore dell'elemento e le sue coordinate.
 * Nella costruzione della lista si richiede di scandire la matrice per colonne.
 * La funzione sparseRow riceve come parametri di ingresso la lista L1
 * restituita dalla funzione sparseList e un indice di riga e restituisce in
 * output una lista L2 contenente soltanto gli elementi di L1 corrispondenti
 * alla riga specificata.
 *
 * Gli elementi della lista sono definiti come segue:
 *
 *  struct node {
 *    int num;
 *    int row;
 *    int col;
 *    struct node *next;
 *  };
 *
 * Esempio: assumendo M = {{0,2,0,0},{1,0,-3,4},{5,0,0,3},{0,0,0,0}},
 * sparseList crea la lista di elementi di tipo struct node
 * (1,1,0) -> (5,2,0) -> (2,0,1) -> (-3,1,2) -> (4,1,3) -> (3,2,3)
 * sparseRow con indice di riga uguale a 2 restituisce
 * la lista di elementi di tipo struct node (5,2,0) -> (3,2,3)
 *
 * Per l'implementazione e la verifica del corretto funzionamento delle funzioni
 * realizzate, si usi lo scheletro di codice sotto riportato e si implementi
 * anche la funzione:
 *
 *  print_list
 *
 * che stampa, per ogni elemento della lista, il valore del campo numero.
 *
 * ============================================================================
 * Per compilare:
 * $ cd   # per spostarsi nella directory HOME (se necessario)
 * $ make
 *
 * Per eseguire il programma:
 * $ ./esercizioCE
 */

#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  char *nome_cognome = NOME_E_COGNOME; // TODO Inserisci nome e cognome
  char *matricola = MATRICOLA;   // TODO Inserisci matricola (es. "0123456")
  printf("%s - Matricola: %s\n", nome_cognome, matricola);

  int nr=4, nc=4;
  int M[4][4] = {{0,2,0,0}, {1,0,-3,4}, {5,0,0,3}, {0,0,0,0}};
```

```
  struct node *list_head1, *list_head2;

  /* Invocare la funzione "sparseList" con la matrice M come argomento */

  print_list(list_head1);

  /* Invocare la funzione "sparseRow" con la lista creata da sparseList come
argomento */

  print_list(list_head2);

  return 0;
}
```

Calcolatori Elettronici A.A. 2019/2020
Prova di laboratorio 19/12/2019 - Turno 2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 * Scrivere le funzioni "sparseList" e "sparseDiag" (ed eventuali funzioni
 * ausiliarie) in C.
 * La funzione sparseList riceve come parametri di ingresso una matrice M sparsa
 * (ovvero una matrice i cui valori sono quasi tutti uguali a 0) di interi,
 * il numero di righe di M ed il numero di colonne di M;
 * la funzione restituisce in output una lista L1 contenente, per ogni elemento
 * di M diverso da zero, il valore dell'elemento e le sue coordinate.
 * Nella costruzione della lista si richiede di scandire la matrice per righe.
 * La funzione sparseDiag riceve come parametro di ingresso la lista L1
 * restituita dalla funzione sparseList e restituisce in output una lista L2
 * contenente soltanto gli elementi di L1 corrispondenti alla diagonale
 * principale della matrice M.
 *
 * Gli elementi della lista sono definiti come segue:
 *
 *  struct node {
 *    int num;
 *    int row;
 *    int col;
 *    struct node *next;
 *  };
 *
 * Esempio: assumendo M = {{0,3,0,0},{0,6,0,-2},{0,0,3,4},{0,0,0,5}},
 * sparseList crea la lista di elementi di tipo struct node
 * (3,0,1) -> (6,1,1) -> (-2,1,3) -> (3,2,2) -> (4,2,3) -> (5,3,3)
 * sparseDiag restituisce la lista di elementi di tipo struct node
 * (6,1,1) -> (3,2,2) -> (5,3,3)
 *
 * Per l'implementazione e la verifica del corretto funzionamento delle funzioni
 * realizzate, si usi lo scheletro di codice sotto riportato e si implementi
 * anche la funzione:
 *
 *  print_list
 *
 * che stampa, per ogni elemento della lista, il valore del campo numero.
 *
 * ============================================================================
 * Per compilare:
 * $ cd   # per spostarsi nella directory HOME (se necessario)
 * $ make
 *
 * Per eseguire il programma:
 * $ ./esercizioCE
 */

#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    char *nome_cognome = NOME_E_COGNOME; // TODO Inserisci nome e cognome
    char *matricola = MATRICOLA;         // TODO Inserisci matricola (es.
"0123456")
    printf("%s - Matricola: %s\n", nome_cognome, matricola);

    int nr=4, nc=4;
```

```
    int M[4][4] = {{0,3,0,0}, {0,6,0,-2}, {0,0,3,4}, {0,0,0,5}};

    struct node *list_head1, *list_head2;

    /* Invocare la funzione "sparseList" con la matrice M come argomento */

    print_list(list_head1);

    /* Invocare la funzione "sparseDiag" con la lista creata da sparseList come
argomento */

    print_list(list_head2);

      return 0;
}
```

Calcolatori Elettronici A.A. 2019/2020
Prova di laboratorio 19/12/2019 - Turno 3

```
/*
 * Scrivere le funzioni "arrayList" e "updateList" (ed eventuali funzioni
 *  ausiliarie) in C.
 * La funzione arrayList riceve come parametri di ingresso un array di interi V
 * e la dimensione di V; la funzione restituisce in output una lista L
 * contenente soltanto gli elementi non nulli dell'array, mantenendo l'ordine
 * degli elementi nell'array.
 * La funzione updateList riceve come parametri di ingresso la lista L, un
 * valore intero value e un indice index ed aggiorna la lista come segue:
 * c1) se index e' l'indice di un nodo presente in L, viene aggiornato il valore
 * del corrispondente nodo con value;
 * c2) se index e' l'indice di un nodo non presente in L, viene inserito un
 * nuovo nodo nella lista con valore pari a value e indice pari a index,
 * mantenendo l'ordine degli elementi nell'array;
 *
 * Gli elementi della lista sono definiti come segue:
 *
 *  struct node {
 *    int val;
 *    int ind;
 *    struct node *next;
 *  };
 *
 * Esempio: assumendo V = {1,0,-1,6,0,4},
 * arrayList crea la lista di elementi di tipo struct node
 * (1,0) -> (-1,2) -> (6,3) -> (4,5)
 * updateList aggiorna la lista come segue:
 * c1) updateList con parametri di ingresso value=7 e index=2 aggiorna la lista
 * in:
 * (1,0) -> (7,2) -> (6,3) -> (4,5)
 * c2) updateList con parametri di ingresso value=8 e index=1 aggiorna la lista
 * in:
 * (1,0) -> (8,1) -> (7,2) -> (6,3) -> (4,5)
 *
 * Per l'implementazione e la verifica del corretto funzionamento delle funzioni
 * realizzate, si usi lo scheletro di codice sotto riportato e si implementi
 * anche la funzione:
 *
 *  print_list
 *
 * che stampa:
 * - per ogni elemento della lista, il valore dei campi val e ind
 * - il massimo, il minimo e la media dei valori presenti nella lista
 *
 * ============================================================================
 * Per compilare:
 * $ cd   # per spostarsi nella directory HOME (se necessario)
 * $ make
 *
 * Per eseguire il programma:
 * $ ./esercizioCE
 */

#include <stdio.h>
#include <stdlib.h>

struct node {
  int val;
  int ind;
  struct node *next;
};
```

```
int main(void)
{
    char *nome_cognome = NOME_E_COGNOME; // TODO Inserisci nome e cognome (es.
"Mario Rossi")
    char *matricola = MATRICOLA;    // TODO Inserisci matricola (es. "0123456")
    printf("%s - Matricola: %s\n", nome_cognome, matricola);

    int len = 6;
    int V[6] = { 1, 0, -1, 6, 0, 4 };

    struct node *list_head;

    /* Invocare la funzione "arrayList" con l'array V come argomento */

    print_list(list_head);

    /* Invocare la funzione "updateList" con la lista creata da arrayList,
    value=7 e index=2 come parametri */

    print_list(list_head);

    /* Invocare la funzione "updateList" con la lista, value=8 e index=1 come
parametri */

    print_list(list_head);

    return 0;
}
```