PR1L - Simulazione

(Tempo a disposizione - 2h)

Dato lo scheletro di codice qui riportato, lo si completi (*esclusivamente sostituendo i puntini come richiesto nei commenti*) aggiungendo l'implementazione delle funzioni/procedure richieste e le chiamate a tali funzioni/-procedure.

bozza.c

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

//List structure:
struct El {
  int  info;
  struct  El *next;
};

typedef struct El  ElementoLista;
typedef ElementoLista *ListaDiElementi;


// Functions/Procedure  to be implemented:

... // add definition of procedure/function readList()
... // add definition of procedure/function cancellaDuplicates()
... // add definition of procedure/function filterLists()

//Function to print all the elements of the list:
void printList(ListaDiElementi list) {
  printf("(");
  while (list != NULL) {
    printf("%d ", list->info);
    list = list->next;
  }
  printf(")\n");
}

int main() {
    ListaDiElementi first_list, second_list;

  //Read and  print the first list:
    ... // add call to procedure/function readList()
```

```
  printf("Prima lista\n");
  printList(first_list);

    //Eliminates  Duplicates from the first list:
    ... // add call to procedure/function  cancellaDuplicates()
    printf("Prima lista senza duplicati\n");
    printList(first_list);



    //Read and  print the second list:
    ...  // add call to procedure/function readList()
    printf("Seconda lista\n");
    printList(second_list);

    //Eliminates  Duplicates from the second list:
    ... // add call to procedure/function  cancellaDuplicates()
    printf("Seconda lista senza duplicati\n");
    printList(second_list);



  //Filter the first list using the elements of the second list:
  ... // add call to procedure/function filterLists()

    //Print the filtered list:
  printf("Lista filtrata\n");
  printList(first_list);

  return 0;
}
```

Le funzioni/procedure da implementare devono rispettare le seguenti specifiche:

- `readList`: Legge dallo standard input una sequenza di numeri interi *ordinati in maniera non descescente* e termina automaticamente l'acquisizione alla prima occorrenza di un numero che non rispetta l'ordinamento (l'intero che viola l'ordinamento non va inserito nella lista). Gli interi devono essere memorizzati, nell'ordine di acquisizione, in una lista concatenata. Per esempio, supponendo che venga acquisita la sequenza $(5, 8, 15, 9)$ la lista di output dovrà assere la seguente:

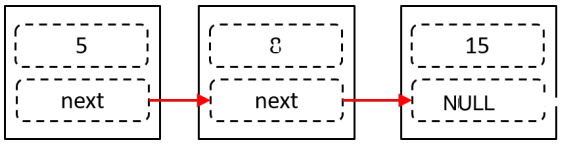

Figura 1: Esempio di acquisizione e memorizzazione di una lista.

- `cancellaDuplicates`: Cancella da una lista tutti i duplicati. *La procedura/funzione deve essere ricorsiva.*
  Per esempio, supponendo che la lista sia $(5, 8, 8, 15, 15, 15)$ la lista modificata dovrà essere quella rappresentata in Figura 2.

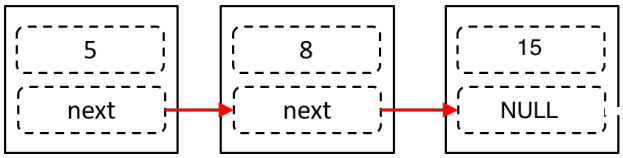

Figura 2: Esempio di una lista senza duplicati.

- `filterList`: date due liste *list1* e *list2* di lunghezza qualsiasi *che non contengono duplicati*, la funzione/procedura elimina dalla lista *list1* tutti gli elementi che sono presenti anche nella lista *list2* . Esempio, date due liste rispettivamente contententi gli interi $(5, 8, 10, 15, 20, 24)$ e $(5, 8, 9, 10, 20, 21)$ la lista risultante dovrà essere quella rappresentata in Figura 3.

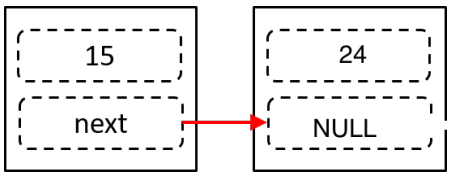

Figura 3: Esempio di prima lista filtrata.

Esempio

Input

```
5
8
15
15
9
5
5
8
9
10
20
20
21
3
```

Output

```
Prima lista
(5 8 15 15 )
Prima lista senza duplicati
(5 8 15 )
Seconda lista
(5 5 8 9 10 20 20 21 )
Seconda lista senza duplicati
(5 8 9 10 20 21 )
Lista filtrata
(15 )
```