Programmazione I e Laboratorio

Prova di laboratorio, Appello III

(Tempo a disposizione - 2h)

Dato lo scheletro di codice qui riportato, lo si completi aggiungendo l'implementazione delle funzioni richieste.

bozza.c

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

//List structure:
struct struct_list_item_t {
  float key;
  struct struct_list_item_t *next;
};

typedef struct struct_list_item_t list_item_t;

//Functions to be implemented:
float *readArray(int *len);
list_item_t *concatInList(float a1[], float a2[], int len1, int
    len2);
list_item_t *orderedDeletion(list_item_t *list);

//Function to print all the elements of the list:
void printList(list_item_t *list) {
  printf("(");
  while (list != NULL) {
    printf("%.1f ", list->key);
    list = list->next;
  }
  printf(")\n");
}

int main() {
  float *A1, *A2;
  int len1, len2, i;
  list_item_t *list1, *list2;

  //Read and print the two arrays:
```

```
  A1 = readArray(&len1);
  printf("Array:\n");
  for (i = 0; i<len1; i++) {
    printf("%.1f ", A1[i]);
  }
  printf("\n");
  A2 = readArray(&len2);
  printf("Array:\n");
  for (i = 0; i<len2; i++) {
    printf("%.1f ", A2[i]);
  }
  printf("\n");

  //Concatenate the two arrays in the list:
  list1 = concatInList(A1, A2, len1, len2);

  //Print the concatenated list:
  printf("Merged list:\n");
  printList(list1);

  //Deletes element out of order in the list
  list2 = orderedDeletion(list1);

  //Print the cleaned list:
  printf("Deleted list:\n");
  printList(list2);

  return 0;
}
```

Le funzioni da implementare devono rispettare le seguenti specifiche:

- `readArray`: Legge dallo standard input una sequenza di numeri in virgola mobile e termina l'acquisizione quando viene letto uno zero. I numeri devono essere memorizzati, nell'ordine di acquisizione, in un array. La funzione restituisce il puntatore al primo elemento dell'array e scrive, nell'indirizzo puntato da *len*, la dimensione dell'array letto. Si può assumere che la sequenza possa prevedere un massimo di 20 elementi.

- `concatInList`: Dati due vettori $a1$ e $a2$ di lunghezza $len1$ e $len2$, la funzione esegue la fusione alternata e localmente ordinata degli elementi delle 2 strutture dati e restituisce il puntatore al primo elemento di una lista l contenente il risultato di tale operazione. In particolare, si scorrano sequenzialmente i 2 array confrontando ad ogni passo gli elementi nella stessa posizione: siano $a1[i]$ e $a2[i]$ gli elementi nella posizione corrente i, questi andranno inseriti in coda alla lista l in ordine crescente di valore, es. $a2[i]$ precederà $a1[i]$ se $a2[i] < a1[i]$ (e viceversa). Gli array possono avere lunghezze differenti: la lunghezza della lista sarà pari al doppio dell'array di lunghezza minore; gli elementi in eccesso presenti nella struttura dati più numerosa andranno

semplicemente *ignorati*. A titolo esemplificativo, si consideri i seguenti 2 array in input

$$a1 = \{1.0, -3.0, 5.0\}$$

$$a2 = \{-2.0, 4.0, 3.0, 1.0, 2.0\},$$

la lista risultante dall'operazione di concatenazione sarà la seguente:

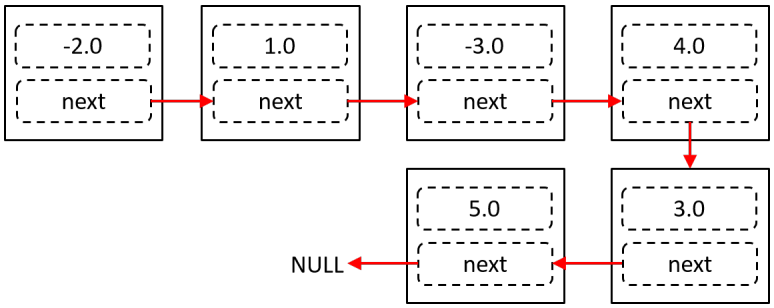

Figura 1: Esempio di concatenazione di due array in lista.

- `orderedDeletion`: prende come parametro di input il puntatore alla testa di una lista l di lunghezza qualsiasi. La funzione scorre la lista dalla testa alla coda eliminando l'elemento corrente se il suo valore è strettamente maggiore $>$ di quello successivo. Ad esempio, sia l la lista rappresentata nella figura precedente (Fig. 1), l'applicazione della funzione `orderedDeletion` ad l genera la lista seguente:

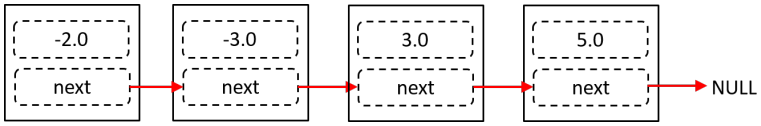

Figura 2: Esempio di cancellazione elementi da lista.

L'elemento in coda alla lista originale, non avendo successore, viene cancellato solo se strettamente minore di zero (< 0). La funzione restituisce il puntatore alla testa della lista modificata.

Esempio

Input

```
1
-3.0
5.0
0
-2.0
4.0
3.0
1.0
2.0
0
```

Output

```
Array:
1.0 -3.0 5.0
Array:
-2.0 4.0 3.0 1.0 2.0
Merged list:
(-2.0 1.0 -3.0 4.0 3.0 5.0 )
Deleted list:
(-2.0 -3.0 3.0 5.0 )
```