Fondamenti di Programmazione e Laboratorio

Prova di laboratorio, 13 febbraio 2018

(Tempo a disposizione 2h)

Dato lo scheletro di codice qui riportato, e disponibile sulla piattaforma come file denominato bozza.c, lo si legga attentamente e lo si completi aggiungendo l'implementazione delle funzioni richieste facendo attenzione a:

- non modificare il main, e in particolare non modificare né aggiungere (in nessuna funzione) comandi di stampa: quelli già presenti in bozza.c garantiscono l'output nel formato richiesto per superare i controlli di correttezza automatici della piattaforma.
- liberare la memoria qualora ve ne sia la possibilità.
- utilizzare le informazioni ricavabili da un'attenta lettura del main (oltre che delle specifiche delle funzioni) per realizzare una soluzione corretta ed efficiente.

È consentito l'uso di ulteriori funzioni ausiliarie, purché opportunamente dichiarate, e commentate per agevolarne la comprensione.

Quando il codice sottoposto alla piattaforma ottiene successo, occorre segnalarlo ai docenti e attendere che gli stessi provvedano alla valutazione della soluzione. Solo con una tale valutazione positiva la prova di laboratorio può considerarsi superata.

bozza.c

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

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

typedef struct El ElementoLista;
```

```c
typedef ElementoLista *ListaDiElementi;

// Functions/Procedure  to be implemented:
ListaDiElementi readList();
int OddEven(ListaDiElementi lista);
ListaDiElementi Equality(ListaDiElementi lista);

//Function that prints 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 list=NULL;
    int res = -1;

    //Read the list
    list=readList();

    //Print the list
    printf("La lista letta e':\n");
    printList(list);

    //call OddEven
    res = OddEven(list);

    //Print the ceiling
    printf("OddEven restituisce: %d\n",res);

    //Print the final list
    printf("La lista troncata e':\n");
    printList(Equality(list));

    return 0;
}
```

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

- `ReadList` legge una sequenza di numeri interi, finché vale la condizione che questa alterni numeri pari e numeri dispari (cominciando indifferentemente per uno o l'altro), e termina l'acquisizione quando legge il primo elemento che viola tale alternanza. I numeri letti devono essere memorizzati in una lista nell'ordine inverso a quello di acquisizione (dunque con inserimento in testa): l'ultimo numero letto, che fa terminare l'acquisizione, **non** va inserito nella lista.
  Per esempio, supponendo che la sequenza immessa sia $(5, 0, 15, 10, 12)$,

al termine di `ReadList` avremmo la lista $(10, 15, 0, 5)$.
Il puntatore alla testa della lista creata deve essere restituito come valore di ritorno della funzione.

- `OddEven`: Questa funzione prende una lista di interi e restituisce 1 se essa contiene lo stesso numero di elementi pari ed elementi dispari. Restituisce 0 altrimenti.
  Per esempio, per la lista $(10, 15, 0, 5)$, `OddEven` restituirà 1. Per la lista $(1, 8, 3)$, `OddEven` restituirà 0.

- `Equality`: data una lista di interi, `Equality` deve cancellare **dalla coda della lista** il minimo numero (che può anche essere = 0) di elementi affinché la lista risultante contenga tanti numeri pari quanti numeri dispari.
  Per esempio, se la lista fosse $(1, 8, 3)$ `Equality` restituirebbe la lista $(1, 8)$.
  Il puntatore alla testa della lista creata deve essere restituito come valore di ritorno della funzione.

Esempio

Input	**Output**

`5`
`0`
`15`
`10`
`11`
`9`
	`La lista letta e':`
	`(11 10 15 0 5 )`
	`OddEven restituisce:  0`
	`La lista troncata e':`
	`(11 10 15 0 )`