Fondamenti di Programmazione con Laboratorio

Prova di Laboratorio del 14 Gennaio 2019

(Tempo a disposizione: 2 ore)

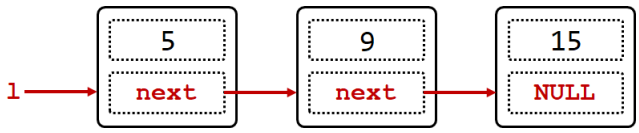
Dato lo scheletro di codice qui riportato (e contenuto nel file `bozza.c`), lo si completi aggiungendo l'implementazione della funzione e della procedura richieste.

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

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

// Function and procedure to be implemented
int read(List *l);
void intersection(List *l1, List l2);

//Function that prints all the elements of the list:
void print(List list) {
   printf("[");
   while (list != NULL) {
      printf("%d ", list->info);
      list = list->next;
   }
   printf("]\n");
}

int main() {
   List list1 = NULL, list2=NULL;
   int size1=0,size2=0;

   //Read list1 and list2
   size1 = read(&list1);
   size2 = read(&list2);

   //Print list1
   printf("Lista 1 (Lunghezza: %d)\n",size1);
   print(list1);

   //Print list2
   printf("Lista 2 (Lunghezza: %d)\n",size2);
   print(list2);

   //Modify list1 to keep its intersection with list2
   intersection(&list1, list2);

   //Print list1 (modified)
   printf("Intersezione\n");
   print(list1);

   return 0;
}
```

Le procedure da implementare devono rispettare la seguente specifica:

`read`: La funzione `read` legge dallo standard input una sequenza di numeri interi *ordinati in maniera strettamente crescente* 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 opportunamente allocata (Nota: Il puntatore `l` —passato come argomento alla procedura— deve essere impostato in modo che punti alla lista creata). La funzione `read` restituisce un intero pari al numero di elementi presenti nella lista creata.

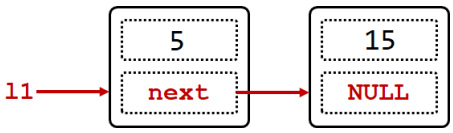
Ad esempio, se viene acquisita la sequenza `[5,9,15,7]` la lista costruita da `read` deve essere la seguente:



e la funzione `read` deve restituire il valore `3`. La stessa lista e lo stesso valore devono essere ottenuti quando la sequenza acquista è `[5,9,15,15]`.

`intersection`: Date due liste `l1` e `l2` di lunghezza qualsiasi, i cui elementi sono *in ordine strettamente crescente* e *che non contengono duplicati*, la procedura `intersection` deve modificare la lista `l1` in modo che mantenga solo gli elementi comuni a entrambe le liste (ovvero eliminando tutti gli elementi della lista `l1` che non sono presenti nella lista `l2`).

Ad esempio, date due liste `l1` e `l2` rispettivamente contenenti `[5, 9, 15]` e `[5, 8, 10, 15, 20, 21]`, la lista `l1` risultante deve essere la seguente:



ATTENZIONE: Il codice già contenuto nel file `bozza.c` non puó essere modificato in alcun modo. L'implementazione di quanto richiesto deve essere aggiunte *dopo* la funzione `main` contenuta nel file `bozza.c`. Inoltre, l'implementazione della procedura `intersection` non deve prevedere l'utilizzo di un array di appoggio o di una terza lista.

Esempio

Input	**Output**

`5`
`9`
`15`
`15`
`5`
`8`
`10`
`15`
`20`
`21`
`3`
	`Lista 1 (Lunghezza: 3)`
	`[5 9 15 ]`
	`Lista 2 (Lunghezza: 6)`
	`[5 8 10 15 20 21 ]`
	`Intersezione`
	`[5 15 ]`