Programmazione I e Laboratorio

Prova di laboratorio, PreAppello A.A. 2018/19

(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.

Non è consentito modificare i prototipi delle funzioni, aggiungendo o modificando parametri.

Non è consentito aggiungere variabili gobali né modificare la definizione delle funzione di stampa.

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>
#define ARR_LEN 30

//Functions to be implemented:
void readbinary(char arr[], int *len, int *reps);
int sum1rec(char arr[], int len);
int existSubseq(char arr[], int len, int reps);

int main() {
  char arr[ARR_LEN];
  int len, reps, i;

  //Read and print the array:
  readbinary(arr, &len, &reps);
  printf("Array:\n");
  for (i = 0; i<len; i++) {
    printf("%d ", arr[i]);
  }
```

```
  printf("\n");

  //Computes how many 1's are occurring (RECURSIVE)
  printf("Totale occorrenze di 1: %d\n", sum1rec(arr,len));

  //Computes if a subsequence of exactly nreps 1's exists in the
     array
  if (existSubseq(arr,len,reps))
    printf("Condizione per %d uno contigui: VERA\n",reps);
  else
    printf("Condizione per %d uno contigui: FALSA\n",reps);
  return 0;
}
```

Le funzioni da implementare devono rispettare le seguenti specifiche:

- `readbinary` legge dallo standard input una sequenza di 0 e 1 e termina l'acquisizione quando viene digitato un numero diverso da 0 o 1. I numeri devono essere memorizzati nell'ordine di acquisizione in un array: l'ultimo numero letto (che fa terminare l'acquisizione) non va inserito nell'array, ma va memorizzato all'indirizzo puntato da *reps*. La funzione scrive, nell'indirizzo puntato da `len`, il numero di elementi inseriti nell'array. Si può assumere che la sequenza possa prevedere un massimo di 30 elementi.

- `sum1rec`: La funzione calcola il numero di 1 presenti all'interno dell'array passato come argomento, la cui lunghezza è passata attraverso il parametro formale `len`, e lo restituisce come valore di ritorno. La funzione DEVE essere implementata in maniera RICORSIVA. L'utilizzo di approcci non ricorsivi all'interno della funzione, o in sue eventuali funzioni ausiliarie, costituisce una violazione della specifica dell'esercizio e porta al non superamento della prova.

- `existSubseq`: La funzione valuta se all'interno dell'array *arr* è presente almeno una sottosequenza di ESATTAMENTE `reps` 1 in posizioni contigue dell'array. Per chiarezza, con *esattamente* si intende che la presenza di quattro 1 contigui nell'array soddisfa la condizione richiesta quando `reps` è pari a 4 ma non quando `reps` è pari, ad esempio, a 2 o 3. La funzione restituisce 1 nel caso in cui la condizione sia verificata e 0 nel caso in cui sia violata.

Esempio

Input	Output
1 1 1 1 0 0 0 1 3	Array: 1 1 1 1 0 0 0 1 Totale occorrenze di 1: 5 Condizione per 3 uno contigui: FALSA