A.A. 2022/2023 - Elaborato 1

Trovare la parola più lunga e la parola più corta all'interno di una frase

Input: un array di BYTE contenente la frase (terminata dal carattere nullo)

Output: due array di BYTE, contenenti la parola più lunga e quella più corta (terminate dal carattere nullo)

Esempi di casi importanti da verificare:

"due parole" → "due", "parole"

"parola" → "parola", "parola"

"a, b, c, test, d," → "a" , "test"

"elaborato 3: 2004" → "3" , "elaborato"

per capire quando finisce devo leggere il carattere nullo

carattere spazio per dividere due parole o caratteri speciali

prima a sinistra in caso di parole con len uguali (scorro da dx a sx comfy)

Scheletro da utilizzare per il programma:

```
/*****************************************************************************
 *                                                                           *
 *                  Architetture dei sistemi di Elaborazione                 *
 *                                                                           *
 *****************************************************************************

 Elaborato 1
 Descrizione:   Data una stringa C (terminata dal carattere nullo), contenente una
           frase (ossia parole separate da spazi e/o segni di punteggiatura),
           trovare la parola più lunga e la parola più corta. In caso di parole
           di uguale lunghezza, considerare la prima da sinistra.
           Le due parole vanno copiate in due array di caratteri come stringhe
           C (terminate dal carattere nullo).
           I segni di punteggiatura da considerare sono: ".,;:'?!"

 *****************************************************************************/


#include <stdio.h>


void main()
{
// Variabili
    #define MAX_LEN 100
    char frase[MAX_LEN] = "Cantami, o Diva, del Pelide Achille l'ira funesta che \
                                                infiniti addusse lutti agli Achei";

    char parolaMax[MAX_LEN+1];
    char parolaMin[MAX_LEN+1];

// Blocco assembler
    __asm
    {



    }

    // Stampa su video
    printf("%s\n%s\n%s\n",frase,parolaMax,parolaMin);
}
```

utilizzare solo la parte degli array che mi serve

A.A. 2022/2023 - Elaborato 2

*Dato in input un numero naturale **n**, restituire i primi **n** termini della successione di Fibonacci.*

Input: una DWORD (il numero **n**)

Output: un array di DWORD (contenente i primi **n** termini della successione di Fibonacci).

Esempi di casi importanti da verificare:

n=0 → {0}

n=1 → {0,1}

n=2 → {0,1,1}

n=3 → {0,1,1,2}

n=12 → {0,1,1,2,3,5,8,13,21,34,55,89,144}

Link utili:

•http://it.wikipedia.org/wiki/Successione_di_Fibonacci

Scheletro da utilizzare per il programma:

```
/*****************************************************************************
 *                                                                           *
 *                  Architetture dei sistemi di Elaborazione                 *
 *                                                                           *
 *****************************************************************************

 Elaborato 2
 Descrizione: Dato in input un numero naturale n, restituire i primi n termini
              della successione di Fibonacci.

 *****************************************************************************/


#include <stdio.h>

void main()
{
    //Variabili
    int n=12;                   //Numero di termini da restituire
    int successione[50];        //Vettore in cui mettere i primi n termini
                                //della successione di Fibonacci

    //Blocco Assembler
    __asm
    {

    }

    //Stampa su video
    {
        int i;
        for(i=0;i<=n;i++)
        {
            printf("%d\n",successione[i]);
        }
    }
}
```

n fornito sempre valido, non serve controllarlo

50 elem: modifico solo le celle che mi servono

non verificare il caso dell'overflow (anche se sarebbe meglio)

A.A. 2022/2023 - Elaborato 3

Dato un array di BYTE, invertire l'ordine dei bit all'interno dell'array.

Input: un array di BYTE e il numero di elementi.

Output: un nuovo array di BYTE, modificato come richiesto.

Per controllare se il risultato del programma è corretto, è indispensabile convertire i numeri in binario.

Esempi di casi da verificare:

{0x00} → {0x00}

{0x01} → {0x80}

{0x01,0x02,0x03} → {0xC0,0x40,0x80}

{0xAA,0xFC,0x09} → {0x90,0x3F,0x55}

3 byte in 24 bit, invertendoli, dovremmo ottenere questa sequenza scorri i byte i lavori sui bit in maniera opportuna, non puoi lavorare direttamente sui bit

Scheletro da utilizzare per il programma:

considerare il vettore come sequenza contigua di bit

```
/*****************************************************************************
 *                                                                           *
 *                    Architetture dei sistemi di Elaborazione               *
 *                                                                           *
 *****************************************************************************

 Elaborato 3
 Descrizione: Dato un array di BYTE, invertire l'ordine dei bit all'interno
              dell'array.

 *****************************************************************************/


#include <stdio.h>

void main()
{
    #define MAX_LEN 100

// Input
    unsigned char vet[]={0xAA,0xFC,0x09};           //Array di BYTE
    unsigned int len=sizeof(vet)/sizeof(vet[0]);    // numero di byte in vet
// Output
    unsigned char res[MAX_LEN];                     //Array di BYTE contenente il risultato

// Blocco assembler
    __asm
    {

    }
// Stampa su video
    {
        unsigned int i;
        for (i=0;i<len;i++)
            printf("res[%2d] = %10d (%08X)\n",i,res[i],res[i]);
    }
```

A.A. 2022/2023 - Consegna

- L'elaborato è da svolgere **singolarmente** e non in gruppo (elaborati uguali o molto simili consegnati da studenti diversi non verranno accettati).
- È necessario corredare il codice sorgente con **commenti** che ne descrivano nel dettaglio il comportamento. commento su ogni riga
- Gli elaborati devono essere consegnati utilizzando il **sistema automatico** presente al link: https://biolab.csr.unibo.it/ElaboratiARC/Home.aspx.
- È bene consegnare un **elaborato** solo quando si è sicuri che **funzioni** correttamente.
- Per potersi iscrivere a un appello d'esame, tutti e 3 gli elaborati devono essere **consegnati** (e valutati corretti - OK) **entro una settimana prima** dalla data dell'appello.
- Per evitare problemi di compilazione in fase di correzione si consiglia di:
  - **Non cambiare** il **nome** delle **variabili** del codice C degli elaborati (attenzione anche alle minuscole/maiuscole).
  - **Non aggiungere** altre **variabili** C al programma, ma usare solo quelle presenti nel testo degli elaborati.
  - **Evitare** di utilizzare **parole inglesi** (es. exit) e lettere accentate nei nomi delle etichette.
- In caso di **problemi** con il sistema di consegna degli elaborati contattare i tutor: dott. Scucchia o dott. Baldini. inizializza i registri a 0, altrimenti non compila

bonus points: numero di sottomissioni, numero di istruzioni, efficienza del codice