AA2011/2012 - Elaborato 1

Cercare tutte le occorrenze di una sottostringa all'interno di una stringa

Input: due vettori di BYTE (le stringhe), due DWORD (le lunghezze delle stringe).

Output: un vettore di DWORD contenente le posizioni in cui è stata trovata la sottostringa. Una DWORD contenente il numero di occorrenze trovate (0=nessuna occorrenza).

Esempi di casi da verificare:

"ciao ciao ciak", "ciao" → {0,5}

"ciao ciao ciak", "cia" → {0,5,10}

"cia", "ciak" → {}

"", "ciao" → {}

Scheletro da utilizzare per il programma:

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

 Elaborato 1
 Descrizione: Cercare tutte le occorrenze di una sottostringa all'interno
              di una stringa

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

#include <stdio.h>

void main()
{
    #define MAX_LEN 100

// Input
    char s1[] = "ciao ciao ciak";
    unsigned int lungS1 = sizeof(s1)-1;
    char s2[] = "ciao";
    unsigned int lungS2 = sizeof(s2)-1;
// Output
    unsigned int posizioni[MAX_LEN];
    unsigned int posizioniLen;

// Blocco assembler
    __asm
    {

    }

// Stampa su video
    {
        unsigned int i;
        for (i=0;i<posizioniLen;i++)
            printf("Sottostringa in posizione=%d\n",posizioni[i]);
    }
}
```

ATTENZIONE:

- Non cambiare il nome delle variabili del codice C scheletro degli elaborati (attenzione anche alle minuscole/maiuscole)
- Non aggiungere altre variabili C al programma, ma usare solo quelle presenti nel testo degli elaborati

AA2011/2012 - Elaborato 2

Generare tutte le permutazioni dei primi N numeri naturali

Input: una DWORD (il numero N)

Output: un vettore di DWORD (tutte le permutazioni, vedi descrizione nel codice a fianco), una DWORD (il numero di permutazioni)

Esempi di casi importanti da verificare:

N=1 Perm={1}, Num=1

N=2 Perm={1,2,2,1}, Num=2

N=4 ...

N=5 ...

Link utili:

•http://en.wikipedia.org/wiki/Permutation

•http://snippets.dzone.com/posts/show/4632

Scheletro da utilizzare per il programma:

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

 Elaborato 2
 Descrizione: Generare tutte le permutazioni dei primi N numeri naturali.
              Le permutazioni generate vanno inserite all'interno di un unico
              array di interi. Ad esempio, se N=3, l'array deve contenere:
              {1,2,3,1,3,2,2,1,3,2,3,1,3,1,2,3,2,1}.

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


#include <stdio.h>


void main() {
// Variabili
int N=4;  // numero di interi (provare con valori <=6)
int Perm[4326];      // array permutazioni: la dimensione è sufficiente per N<=6
int Num;  // Alla fine deve contenere il numero di permutazioni generate

// Blocco assembler
     __asm {

     }

// Stampa su video
     {
          int i,j,k;
          printf("Permutazioni dei primi %d numeri naturali\n",N);
          for (i=k=0;i<Num;i++) {
               for (j=0;j<N;j++) {
                    printf("%3d",Perm[k++]);
               }
               printf("\n");
          }
     }
}
```

ATTENZIONE:

- Non cambiare il nome delle variabili del codice C scheletro degli elaborati (attenzione anche alle minuscole/maiuscole)
- Non aggiungere altre variabili C al programma, ma usare solo quelle presenti nel testo degli elaborati

AA2011/2012 - 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}

Scheletro da utilizzare per il programma:

```
/*****************************************************************************
 *                                                                           *
 *                  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]);
    }
}
```

ATTENZIONE:

- Non cambiare il nome delle variabili del codice C scheletro degli elaborati (attenzione anche alle minuscole/maiuscole)
- Non aggiungere altre variabili C al programma, ma usare solo quelle presenti nel testo degli elaborati