AA2005/2006 - Elaborato 1

Sostituire tutte le occorrenze di una sottostringa all'interno di una stringa.
Ad esempio ("Questa stringa è una stringa","stringa","mela") → "Questa mela è una mela"

Input: la stringa in cui effettuare la sostituzione, la stringa da cercare e quella con cui sostituirla (tutti array di BYTE terminanti con 0)

Output: la stessa stringa modificata

Esempi di casi importanti da verificare:

Sostituzione dell'intera stringa

Nuova stringa da sostituire più corta o più lunga

Stringa da sostituire con lunghezza 0

N.B. La stringa modificata deve essere terminata da un carattere nullo (byte 0) esattamente come le stringhe di input.

Scheletro da utilizzare per il programma:

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

 Elaborato 1
 Descrizione: Sostituire tutte le occorrenze di una sottostringa all'interno di una stringa.
              Ad esempio ("Questa stringa è una stringa","stringa","mela") ->
               "Questa mela è una mela".
               Ogni stringa termina con il carattere nullo (codice ASCII 0).

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

#include <stdio.h>

void main() {
    // La stringa da modificare
    char stringa[1024] = "Questa stringa e' una stringa";
    char str1[] = "stringa";     // la sottostringa da cercare
    char str2[] = "mela";    // la sottostringa con cui sostituirla

    __asm {


    }

    // Stampa su video
    printf("Nuova stringa: %s\n",stringa);
}
```

Si supponga che il buffer di memoria cui corrisponde la stringa da modificare sia sempre sufficientemente ampio per il risultato.

AA2005/2006 - Elaborato 2

Dato un punto (x,y) nel piano, trovare il punto più vicino ad esso fra tutti quelli contenuti in un insieme di n punti. *Ogni punto è rappresentato da una DWORD: la WORD meno significativa è la coordinata x, la più significativa la y.*

Input: una DWORD (il punto), un array di DWORD (l'insieme dei punti), una DWORD (numero di punti nell'insieme)

Output: una DWORD (indice nell'array del punto più vicino).

Esempi di casi importanti da verificare:

Insieme contenente un solo punto

Alcuni punti con una o entrambe le coordinate negative

N.B. Le coordinate dei punti possono essere sia positive che negative (in complemento a due), comprese nell'intervallo [-1000,1000].

Scheletro da utilizzare per il programma:

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

 Elaborato 2
 Descrizione: Dato un punto (x,y) nel piano, trovare il punto più vicino ad esso
              fra tutti quelli contenuti in un insieme di n punti.
              Ogni punto è rappresentato da una DWORD: la WORD meno
              significativa è la coordinata x, la più significativa la y.
 *****************************************************************************/

#include <stdio.h>


void main() {

    unsigned int Point = (5<<16 | 3); // il punto
    unsigned int PointSet[] = { (10<<16) | 3, (4<<16) | 2, (5<<16) | 20 };  // insieme
    unsigned int n = sizeof(PointSet)/sizeof(PointSet[0]);  // numero punti nell'insieme
    unsigned int index; // risultato: indice del punto piu' vicino a Point


    __asm {
    }

    // Stampa su video
    printf("Il punto più vicino a (%d,%d) e' (%d,%d) [indice=%d]\n",
        (short int)(Point&0xFFFF),(short int)(Point>>16),
        (short int)(PointSet[index]&0xFFFF),(short int)(PointSet[index]>>16),
        index
    );

}
```

AA2005/2006 - Elaborato 3

Data una sequenza di bit, sapendo che ogni n bit (di dati) vi è un bit di parità (1 se il numero di bit a 1 fra i precedenti n è dispari), verificare se vi sono errori.

Input: un'array di BYTE (da considerare come una sequenza di bit), una DWORD (la lunghezza della sequenza di bit), un BYTE (il valore *n*)

Output: un BYTE (contenente 1 se vi sono errori, altrimenti 0)

N.B. viene specificata in input la lunghezza *in bit*: può quindi accadere che l'ultimo BYTE dell'array sia utilizzato solo in parte.

Si può supporre che la lunghezza della sequenza di bit sia sempre multipla di (*n*+1).

Scheletro da utilizzare per il programma:

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

 Elaborato 3
 Descrizione:   Data una sequenza di bit, sapendo che ogni n bit (di dati) vi è
                un bit di parità (1 se il numero di bit a 1 fra i precedenti n è
                dispari), verificare se vi sono errori.

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


#include <stdio.h>


void main() {
// Variabili
    unsigned char vet[]={2,4,67,2,2,58,99}; // sequenza di bit
    unsigned int len = 55;  // lunghezza (numero di bit)
    unsigned char n = 4;    // numero di bit dati
    unsigned char errori=0; // risultato (1=errori, 0=no errori)

    __asm {

    }

    // Stampa su video
    printf("La sequenza di bit %scontiene errori\n",(errori?"":"non "));

}
```