[T07] Esercitazione 7

Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ciascun esercizio di programmazione. Non modificate in alcun modo i programmi di test `*_main.c`.
- Rinominare la directory chiamandola `cognome.nome`. Sulle postazioni del laboratorio sarà `/home/biar/Desktop/cognome.nome/`.
- È possibile consultare appunti/libri e il materiale didattico online.
- Rispondete alle domande online sul modulo di consegna.
- **Finiti gli esercizi**, e non più tardi della fine della giornata:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione).
  - fate **upload** del file `cognome.nome.zip`.
  - **importante**: fate logout dal vostro account Google!
- **Se avete domande** accedete a Zoom agli orari stabiliti per l'esercitazione, accedendo con la **mail istituzionale** uniroma1.it dello studente. Troverete online i docenti ed il tutor del corso. In alternativa, scrivete via mail ai docenti.
- **Suggerimento** Non traducete direttamente da C a IA32, ma scrivere prima una versione C intermedia `E*/e*_eq.c` equivalente a quella di partenza, ma più semplice da tradurre in assembly. Testatela con il main di prova prima di passare a scrivere la versione `.s`. E' inutile tradurre la versione C equivalente se è errata!

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

Esercizio 1 (Ordinamento e ricerca su array di strutture)

Si vuole scrivere nel file `E1-find-person/e1.c` due funzioni che lavorano su array di strutture che rappresentano persone come definito di seguito:

```
typedef struct person_t person_t;

struct person_t {
        char *name;
        int age;
};
```

La prima funzione ha il seguente prototipo:

```
void sort_people(person_t p[], size_t n);
```

e deve ordinare per nome le persone nell'array `p` di dimensione `n`. La seconda funzione ha il seguente prototipo:

```
person_t *find_person(char *key, person_t sorted[], size_t n);
```

e, assumendo che `sorted` sia un array di `n` persone ordinate per nome e `key` è un nome da cercare, restituisce il puntatore a una persona nell'array che ha quel nome, oppure `NULL` se nessuna persona ha quel nome. Se vi sono più persone con il nome cercato, restituisce una qualunque.

Usare il main di prova nella directory di lavoro `E1-find-person` compilando con `gcc e1_main.c e1.c -o e1`.

Esercizio 2 (Concatenazione di liste collegate)

Tradurre in IA32 nel file `E2-list-concat/e2.s` la seguente funzione C contenuta in `E2-list-concat/e2.c` che, data una lista `*l1` ne appende un'altra `l2` in coda. E' possibile che sia `*l1` che `l2` siano vuote. Usare il file `E2-list-concat/e2_eq.c` per sviluppare la versione C equivalente.

```c
#include <stdlib.h>
#include "e2.h"

void list_concat(node_t **l1, node_t *l2) {
    node_t *p = *l1;
    if (p==NULL) *l1 = l2;
    else {
        while (p->next!=NULL) p = p->next;
        p -> next = l2;
    }
}
```

Usare il main di prova nella directory di lavoro `E2-list-concat` compilando con `gcc -m32 e2_main.c e2.s -o e2`.

Esercizio 3 (Tokenizzazione di stringhe)

Si vuole sviluppare una funzione che estrae i token di una stringa e li restituisce come array di stringhe. Scrivere nel file `E3-tokenizer/e3.c` un'implementazione di una funzione `tokenize` con il seguente prototipo:

```c
char **tokenize(const char *str, const char *delim, int *num_tok);
```

Dove `str` è una stringa in sola lettura, `delim` è una stringa in sola lettura che contiene dei delimitatori e `num_tok` è l'indirizzo di una variabile in cui scrivere il numero di token trovati in `str`. La funzione restituisce un array di stringhe allocato dinamicamente che contiene i token trovati, anch'essi allocati dinamicamente.

Suggerimento: si veda la funzione `deallocate_token_list` in `e3_main.c` per comprendere come deve essere allocato l'array di token restituito da `tokenize`.

Usare il main di prova nella directory di lavoro `E3-tokenizer` compilando con `gcc e3_main.c e3.c -o e3`.

Esercizio 4 (Domande)

Rispondi alle seguenti domande, tenendo conto che una risposta corretta vale 1 punti, mentre una risposta errata vale 0 punti.

Domanda 1. Si consideri un programma che usa la funzione matematica `sin`. Quale delle seguenti righe di comando potrebbe essere utilizzata?

- A. gcc main.c -math -o mymath
- B. gcc main.c -o mymath
- C. gcc main.c -o mymath -lm
- D. nessuna delle precedenti

Domanda 2. Si consideri una funzione comparatore `int compar(const void *a, const void *b)`. Quale delle seguenti affermazioni è falsa?

- A. se l'elemento puntato da a è minore di quello puntato da b la funzione restituisce un valore negativo
- B. se l'elemento puntato da a è minore o uguale di quello puntato da b la funzione restituisce un valore negativo
- C. se l'elemento puntato da a è maggiore di quello puntato da b la funzione restituisce un valore positivo
- A. se l'elemento puntato da a è uguale a quello puntato da b la funzione restituisce un valore zero

Domanda 3. Si consideri la seguente istruzione `fseek(f,0,SEEK_CUR)`. Quale delle seguenti affermazioni è vera?

- A. l'istruzione sposta la posizione corrente del file `f` all'inizio del file
- B. l'istruzione sposta la posizione corrente del file `f` alla fine del file
- C. l'istruzione non sposta la posizione corrente del file `f`
- D. l'istruzione non è valida

Domanda 4. Si consideri la seguente istruzione `sscanf(s,"%d %d %d", &a, &b, &c)`. Quale delle seguenti affermazioni è vera?

- A. l'istruzione restituisce il numero di byte letti dalla stringa `s`
- B. l'istruzione restituisce zero se riesce a leggere tutti gli argomenti `&a, &b, &c`
- C. l'istruzione restituisce il numero di argomenti letti
- D. l'istruzione restituisce il numero di argomenti letti che assumono un valore positivo

Soluzioni

Esercizio 1 (Ordinamento e ricerca su array di strutture)

```c
#include "e1.h"
#include <string.h>
#include <stdlib.h>

int cmp(const void *ap, const void *bp){
        person_t a = *(person_t*)ap;
        person_t b = *(person_t*)bp;
        return strcmp(a.name, b.name);
}

void sort_people(person_t p[], size_t nel) {
        qsort(p, nel, sizeof(person_t), cmp);
}

person_t *find_person(char *key, person_t sorted[], size_t nel) {
        person_t key_person;
        key_person.name = key;
```

```
        return bsearch(&key_person, sorted, nel, sizeof(person_t), cmp);
}
```

Esercizio 2 (Concatenazione di liste collegate)

```
.globl list_concat
                                # regs allocation: l1==edx, *li==eax, l2==ecx
list_concat:                    # void list_concat(node_t **l1, node_t *l2) {
    movl 8(%esp), %ecx
    movl 4(%esp), %edx          # node_t **tmp = l1;
    movl (%edx), %eax           # node_t *p = *tmp;
    cmpl $0, %eax               # if (p!=NULL)
    jnz L                       #     goto L;
    movl %ecx, (%edx)           # *l1 = l2;
    jmp E                       # goto E;
L:  cmpl $0, 4(%eax)            # if ((*p).next==NULL)
    jz T                        #   goto T;
    movl 4(%eax), %eax          # p = (*p).next;
    jmp L                       # goto L;
T:  movl %ecx, 4(%eax)          # (*p).next = l2;
E:  ret                         # return
```

Esercizio 3 (Tokenizzazione di stringhe)

```
#include "e3.h"
#include <string.h>
#include <stdlib.h>

char **tokenize(const char *str, const char *delim, int *num_tok){
        char *buf = malloc(strlen(str)+1);
        strcpy(buf, str);
        char *token = strtok(buf, delim);
        int cnt = 0;
        while (token != NULL) {
                cnt++;
                token = strtok(NULL, delim);
        }
        char **res = malloc(cnt*sizeof(char*));
        strcpy(buf, str);
        token = strtok(buf, delim);
        int i = 0;
        while (token != NULL) {
                res[i] = malloc(strlen(token)+1);
                strcpy(res[i], token);
                i++;
                token = strtok(NULL, delim);
        }
        free(buf);
        *num_tok = cnt;
        return res;
}
```

Esercizio 4 (Domande)

1. C. gcc main.c -o mymath -lm
2. B. se l'elemento puntato da a è minore o uguale di quello puntato da b la funzione restituisce un valore negativo

3. C. l’istruzione non sposta la posizione corrente del file f
4. C. l’istruzione restituisce il numero di argomenti letti