Esame di Programmazione I, 24 giugno 2013. 2 ore

Esercizio 1 [**13 punti**] Si scriva una funzione

```
char *somma_modulo_10(const char *s)
```

che restituisce la somma delle cifre contenute in `s`, modulo 10. La somma deve essere restituita come una nuova stringa. Per esempio, un'esecuzione del seguente programma:

```
int main(void) {
  char buffer[101];
  char *s;

  scanf("%100s", buffer);
  printf("somma delle cifre modulo dieci = %s\n", s = somma_modulo_10(buffer));
  free(s)

  return 0;
}
```

deve essere la seguente:

```
Ciao2013Oggi67Domani34Hello                  <- input immesso dall'utente
somma delle cifre modulo dieci = sei         <- risposta del calcolatore
```

Esercizio 2 [**9 punti**] Si considerino le liste di caratteri viste a lezione. Si implementi una funzione ricorsiva:

```
int alternata(struct list *this)
```

che determina se la lista `this` è fatta da un'alternanza di singole vocali (minuscole o maiuscole) e di singoli caratteri non vocali. Se tutto è corretto, l'esecuzione del programma:

```
#include <stdio.h>
#include "list.h"

int main(void) {
  struct list *l1 = construct_list
    ('a', construct_list('f', construct_list('I', construct_list('*', construct_list('e', NULL)))));
  struct list *l2 = construct_list('a', construct_list('f', construct_list('&', NULL)));

  printf("l1 = "); print_list(l1); printf("\n");
  if (alternata(l1))
    printf("l1 e' alternata\n");
  else
    printf("l1 non e' alternata\n");

  printf("l2 = "); print_list(l2); printf("\n");
  if (alternata(l2))
    printf("l2 e' alternata\n");
  else
    printf("l2 non e' alternata\n");

  return 0;
}
```

dovrà stampare:

```
l1 = [a, f, I, *, e]
l1 e' alternata
l2 = [a, f, &]
l2 non e' alternata
```

Esercizio 3 [9 punti] Le note musicali sono normalmente divise in 12 *semitoni*, come nella seguente tabella:

semitono	nome italiano	nome inglese
0	do	C
1	do#	C#
2	re	D
3	re#	D#
4	mi	E
5	fa	F
6	fa#	F#
7	sol	G
8	sol#	G#
9	la	A
10	la#	A#
11	si	B

Si scrivano i file `note.h` e `nota.c` in modo da definire una struttura `nota` con le seguenti funzioni:

```
struct nota *construct_nota(int semitono);
void destroy_nota(struct nota *this);
void print_nota_it(struct nota *this);
void print_nota_uk(struct nota *this);
void incrementa_nota(struct nota *this, int inc);
struct nota *nota_incrementata(struct nota *this, int inc);
```

dove si assume che `construct_nota` riceva sempre un numero tra 0 e 11; `print_nota_it` stampa il nome italiano della nota `this` mentre `print_nota_uk` ne stampa il nome inglese; `incrementa_nota` incrementa di `inc` il semitono della nota `this` (in modo ciclico: dopo il "si" ricominciamo dal "do"); `nota_incrementata` restituisce una *nuova* nota ottenuta incrementando di `inc` il semitono di `this` (che non viene modificato).

Se tutto è corretto, l'esecuzione del seguente programma:

```
#include <stdio.h>
#include "note.h"

int main(void) {
  struct nota *n1 = construct_nota(5); struct nota *n2 = construct_nota(11); struct nota *n3;

  printf("n1: "); print_nota_it(n1); printf(" ");
  print_nota_uk(n1); printf("\n");
  printf("n2: "); print_nota_it(n2); printf(" ");
  print_nota_uk(n2); printf("\n");

  incrementa_nota(n1, 11);
  n3 = nota_incrementata(n2, 23);

  printf("n1: "); print_nota_it(n1); printf(" ");
  print_nota_uk(n1); printf("\n");
  printf("n2: "); print_nota_it(n2); printf(" ");
  print_nota_uk(n2); printf("\n");
  printf("n3: "); print_nota_it(n3); printf(" ");
  print_nota_uk(n3); printf("\n");

  destroy_nota(n1); destroy_nota(n2); destroy_nota(n3);
  return 0;
}
```

deve stampare:

```
n1: fa F
n2: si B
n1: mi E              <- n1 e' stato modificato
n2: si B              <- n2 non e' stato modificato!
n3: la# A#
```