Esame di Programmazione I, 4 febbraio 2013. 2 ore

Esercizio 1 [11 punti] Si scriva una funzione

```
char *mirror(const char *s)
```

che restituisce una nuova stringa ottenuta riflettendo `s` rispetto al suo ultimo carattere. Per esempio, l'esecuzione del seguente programma:

```
int main(void) {
  char *buffer;

  printf("%s\n", buffer = mirror("ciao")); free(buffer);
  printf("%s\n", buffer = mirror("che bella giornata!")); free(buffer);
  printf("%s\n", buffer = mirror("")); free(buffer);

  return 0;
}
```

deve stampare:

```
ciaoaic
che bella giornata!atanroig alleb ehc
                                          <- qui c'e' la stringa vuota!
```

Esercizio 2 [10 punti] Si considerino le liste viste a lezione. Si implementi una funzione ricorsiva:

```
struct list *lastToFirst(struct list *this)
```

che restituisce una lista uguale a `this` tranne l'ultimo elemento di `this`, che nella lista risultante si trova spostato all'inizio. La lista `this` non deve essere modificata. Se tutto è corretto, l'esecuzione del programma:

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

int main(void) {
  struct list *l1 = construct_list(10, construct_list(-5, construct_list
   (23, construct_list(11, construct_list(-2, NULL)))));

  struct list *l2 = construct_list(10, NULL);
  struct list *l3 = NULL;

  printf("l1 = ");
  print_list(l1);
  printf("\n");

  printf("lastToFirst(l1) = ");
  print_list(lastToFirst(l1));
  printf("\n");

  printf("l1 = ");
  print_list(l1);
  printf("\n");

  printf("lastToFirst(l2) = ");
  print_list(lastToFirst(l2));
  printf("\n");

  printf("lastToFirst(l3) = ");
  print_list(lastToFirst(l3));
  printf("\n");

  return 0;
}
```

dovrà stampare:

```
l1 = [10, -5, 23, 11, -2]
lastToFirst(l1) = [-2, 10, -5, 23, 11]
l1 = [10, -5, 23, 11, -2]
lastToFirst(l2) = [10]
lastToFirst(l3) = []
```

Esercizio 3 [10 punti] Si definisca una struttura `television` che implementa una televisione, sintonizzabile su un insieme di canali. Si scrivano i file `television.h` e `television.c` implementando le funzioni:

- `struct television *construct_television(const char *names[], int num_channels)` che restituisce una nuova televisione con `num_channel` canali chiamati come indicato dall'array, che si assume che sia di lunghezza `num_channel`; la televisione nasce sintonizzata sul canale numero 0;
- `void destruct_television(struct television *this)`, che dealloca la televisione `this`;
- `void set_channel(struct television *this, int channel)`, che sintonizza la televisione sul canale numero `channel`, che si assume tra 0 (incluso) e il numero di canali della televisione (escluso);
- `void undo(struct television *this)`, che torna indietro nella storia, cioè sintonizza la televisione sul canale su cui era sintonizzata prima di quello corrente. Se si torna fino all'inizio della storia, questo metodo non fa nulla;
- `char *toString(struct television *this)`, che restituisce il nome del canale su cui la televisione `this` è in questo momento sintonizzata.

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

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

int main(void) {
  const char *names[] = { "raggi zero", "canale uno", "sei sul due" };

  struct television *t = construct_television(names, 3);

                           printf("inizio:            %s\n", toString(t));
  set_channel(t, 2);       printf("set_channel(t, 2): %s\n", toString(t));
  set_channel(t, 1);       printf("set_channel(t, 1): %s\n", toString(t));
  set_channel(t, 2);       printf("set_channel(t, 2): %s\n", toString(t));
  undo(t);                 printf("undo(t):           %s\n", toString(t));
  undo(t);                 printf("undo(t):           %s\n", toString(t));
  undo(t);                 printf("undo(t):           %s\n", toString(t));
  undo(t);                 printf("undo(t):           %s\n", toString(t));
  set_channel(t, 2);       printf("set_channel(t, 2): %s\n", toString(t));
  undo(t);                 printf("undo(t):           %s\n", toString(t));

  destruct_television(t);

  return 0;
}
```

deve stampare:

```
inizio:            raggi zero
set_channel(t, 2): sei sul due
set_channel(t, 1): canale uno
set_channel(t, 2): sei sul due
undo(t):           canale uno
undo(t):           sei sul due
undo(t):           raggi zero
undo(t):           raggi zero              <- qui la storia e' ormai finita e undo(t) non fa nulla
set_channel(t, 2): sei sul due
undo(t):           raggi zero
```