COGNOME: **NOME:** **MATRICOLA:**

Esame di Programmazione I, 11 luglio 2012. 2 ore

Esercizio 1 [**12 punti**] Si scriva una funzione

```
char *replace(const char *s, const char *old, const char *new)
```

che restituisce una nuova stringa ottenuta sostituendo con `new` la prima occorrenza (quella più a sinistra) di `old` dentro `s`. Se tutto è corretto, l'esecuzione del seguente programma:

```
#include <stdio.h>

int main(void) {
  const char *x = "ciao, come stai?";
  char *r = replace(x, "come stai", "how are you");
  printf("%s\n", r);
  free(r);
  r = replace(x, "a", "e");
  printf("%s\n", r);
  free(r);
  r = replace(x, "", "_inizio_");
  printf("%s\n", r);
  free(r);
  r = replace(x, "hello", "ciao");
  printf("%s\n", r);
  free(r);
  return 0;
}
```

deve stampare:

```
ciao, how are you?
cieo, come stai?
_inizio_ciao, come stai?
ciao, come stai?
```

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

```
int prefix(struct list *this, struct list *other);
```

che restituisce *vero* se e solo se `other` è un prefisso di `this` (cioè se e solo se `this` comincia con `other`). Si noti che la lista vuota è un prefisso di qualsiasi altra lista.

Se tutto è corretto, l'esecuzione del programma:

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

int main(void) {
  struct list *l = construct_list(11, construct_list(13, construct_list
      (-4, construct_list(13, construct_list(-6, construct_list(0, construct_list(1, NULL)))))));

  struct list *p1 = construct_list(11, construct_list(13, NULL));
  struct list *p2 = construct_list(13, construct_list(-4, construct_list(13, NULL)));

  print_list(p1);
  if (!prefix(l, p1))
    printf(" non");
  printf(" e' un prefisso di ");
  print_list(l);
  printf("\n");
```

```
  print_list(p2);
  if (!prefix(l, p2))
    printf(" non");
  printf(" e' un prefisso di ");
  print_list(l);
  printf("\n");

  return 0;
}
```

dovrà stampare:

```
[11, 13] e' un prefisso di [11, 13, -4, 13, -6, 0, 1]
[13, -4, 13] non e' un prefisso di [11, 13, -4, 13, -6, 0, 1]
```

Esercizio 3 [11 punti] Si definisca una struttura `radio` che implementa una radio con 10 canali (numerati da 0 a 9). Si scrivano i file `radio.h` e `radio.c` implementando le funzioni:

- `struct radio *construct_radio()` che restituisce una nuova radio, i cui dieci canali non sono ancora memorizzati e che è sintonizzata sul canale 0;
- `void destruct_radio(struct radio *this)` che dealloca la radio `this`;
- `void memorizza_canale(struct radio *this, int numero, char *nome, float frequenza)`, che memorizza il canale di numero indicato di `this` sull'emittente di nome `nome` la cui frequenza è `frequenza`;
- `void sintonizza_su(struct radio *this, int numero)`, che sintonizza la radio `this` sul canale di numero indicato;
- `char *toString(struct radio *this)`, che restituisce una nuova stringa del tipo

  ```
  Radio Arena: frequenza 102.5
  ```

  (il nome dell'emittente seguito dall'indicazione della frequenza). Se la radio è sintonizzata su un canale non ancora memorizzato, questa funzione deve restituire la nuova stringa `canale non memorizzato`.

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

```
#include <stdlib.h>
#include <stdio.h>
#include "radio.h"

int main(void) {
  struct radio *r = construct_radio();
  char *s = toString(r);
  printf("%s\n", s);
  free(s);
  memorizza_canale(r, 3, "Radio Patchanka", 98.1);
  memorizza_canale(r, 0, "Radio Castelrotto", 100.5);
  memorizza_canale(r, 9, "Radiofutura", 105.6);
  sintonizza_su(r, 3);
  printf("%s\n", s = toString(r));
  free(s);
  memorizza_canale(r, 0, "Radio Arena", 102.5);
  sintonizza_su(r, 0);
  printf("%s\n", s = toString(r));
  free(s);
  destruct_radio(r);
  return 0;
}
```

deve stampare:

```
canale non memorizzato
Radio Patchanka: frequenza 98.1
Radio Arena: frequenza 102.5
```