Ingegneria degli Algoritmi (A.A. 2014-2015)

Corsi di Laurea in Ingegneria Informatica e Automatica, Ingegneria dei Sistemi Informatici, e Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

Esame di laboratorio (17/06/2015) – Durata 2h 30'

Realizzazione di una coda generica con lista collegata

Si richiede di completare le funzioni di una coda di elementi generici relizzata mediante lista collegata in C, con i seguenti prototipi dichiarati nella header `queue.h`:

```
queue* queue_new(size_t elem_size);
```

Crea una coda vuota:

1. `elem_size`: numero di byte di un elemento della coda;
2. la funzione deve restituire il puntatore alla coda allocata.

```
void queue_delete(queue* q);
```

Dealloca una coda:

1. `q`: puntatore alla coda da deallocare.

```
int queue_isempty(const queue* q);
```

Verifica se la coda è vuota:

1. `q`: puntatore alla coda;
2. la funzione deve restituire 1 se la coda è vuota e 0 altrimenti.

```
void queue_enqueue(queue* q, void* elem);
```

Aggiunge un elemento alla fine della coda:

3. `q`: puntatore alla coda;
4. `elem`: puntatore a un oggetto che contiene l'elemento da inserire in coda.

```
int queue_dequeue(queue* q, void* elem);
```

Toglie un elemento dall'inizio della coda:

1. `q`: puntatore alla coda;
2. `elem`: puntatore a un buffer in cui scrivere l'elemento rimosso dalla coda, oppure `NULL` se l'elemento va ignorato;
3. la funzione deve restituire -1 se la coda è vuota e 0 altrimenti.

```
int queue_first(const queue* q, void* elem);
```

Fornisce l'elemento all'inizio della coda senza toglierlo (`const` sta ad indicare che la coda non viene modificata):

1. `q`: puntatore alla coda;
2. `elem`: puntatore a un buffer in cui scrivere l'elemento rimosso dalla coda;

3. la funzione deve restituire -1 se la coda è vuota e 0 altrimenti.

```
void queue_clear(queue* q);
```

Rimuove tutti gli elementi da una coda:

- `q`: puntatore alla coda da svuotare.

Scrivere le funzioni nel file `queue.c`. E' utile esaminare il main di prova `main.c` per vedere un esempio di operazioni su una pila.

Compilazione e test.

- *Compilazione programma di test*: dare il comando `make`, che genera il file eseguibile `queue` contenente un programma di test
- *Test correttezza*: eseguire il programma di test `./queue`.

E' consigliabile modificare il main per fare ulteriori test e verificare la correttezza delle funzioni scritte, purché non si modifichino i prototipi delle funzioni dichiarate in `queue.h`.

Nota bene: testare la correttezza dell'uso della memoria usando il tool Valgrind:

`valgrind ./queue`