Fondamenti di Informatica
Esempio programmazione 2

Istruzioni per lo svolgimento e la consegna

- La prima operazione da effettuare è modificare il file `studente.txt` (presente nella directory dove avete trovato questo `pdf`) inserendo il proprio nome e cognome e numero di matricola. Utilizzare un semplice editor di testo, salvando il file senza modificarne il nome.
- Nella stessa directory sono presenti i file necessari allo svolgimento dell'esercizio. **Per ogni esercizio dovrà essere modificato solamente il file `.c` corrispondente. Non devono essere modificati né spostati o eliminati i rimanenti file, pena la valutazione negativa dell'assegnamento**. Nel file `.c` dovranno essere mantenuti tutti e soli gli output a schermo forniti, in modo da mantenere la corrispondenza con l'output di esempio. Inoltre, è possibile definire funzioni ausiliarie all'interno del sorgente `.c`.
- Per compilare e generare l'eseguibile, da terminale entrate nella directory dove avete trovato questo `pdf` e lanciate il comando `make nome_esercizio`. Il `nome_esercizio` corrisponde al nome del sorgente, privato dell'estensione `.c`. Verrà generato l'eseguibile `nome_esercizio` che, lanciato da terminale (`./nome_esercizio`), vi permetterà di provare il vostro programma.
- Lanciando invece il comando (`./self_evaluation nome_esercizio`) eseguirete in maniera automatica alcuni test per verificare le soluzioni che avete implementato. I test sono studiati per verificare anche i casi particolari, in modo da gestire quelli che possono essere errori comuni in fase di implementazione. **Tenete presente che il correttore funziona solo all'interno di una distribuzione Linux a 64 bit (ad esempio, le macchine messe a disposizione nel laboratorio)**.
- La procedura di consegna dovrà iniziare lanciando il programma `./prepara_consegna.sh` presente nella directory dove avete trovato il presente `pdf`. Una volta lanciato, esso genererà un archivio di nome `consegna.tar.gz`: tale file sarà **l'unico che dovrà essere inviato** attraverso il sito `https://stem.elearning.unipd.it` per consegnare il vostro elaborato, seguendo anche le istruzioni che saranno fornite in aula dai docenti.

Considerate 2 aspetti:

- Se ci sono errori in compilazione/esecuzione, c'è qualcosa che rende errata/incompleta la vostra implementazione;
- Se non ci sono errori in compilazione/esecuzione, verificate che i risultati siano corretti (in alcuni casi è molto semplice fare il calcolo anche a mente).

1 Liste selezionate (`list_select.c`)

Data una lista concatenata di `int`, sviluppare e testare la seguente funzione C:

```
// Crea una nuova lista composta dai dati contenuti nella lista
// puntata da data_head che si trovano nelle posizioni fornite
// tramite la lista puntata dal pos_head
struct Node* list_select(struct Node *data_head, struct Node *pos_head);
```

Completare la funzione sapendo che:

- `data_head` è il puntatore alla lista che contiene i dati da esaminare all'interno della funzione;
- `pos_head` è il puntatore alla lista che contiene le posizioni dei dati da inserire in una nuova lista;
- la nuova lista deve essere allocata dinamicamente;
- la nuova lista avrà tanti nodi quanti la lista delle posizioni;
- i nodi della nuova lista saranno creati a partire dai dati presenti nella lista puntata da `data_head` alle posizioni contenute nella lista puntata da `pos_head`;
- se una delle due liste è vuota ritornare `NULL`;
- le posizioni fornite sono comunque valide per i dati in input e non serve implementare controlli per verificare che tale vincolo sia rispettato.

Il programma di test è già implementato e compilato come file oggetto `list_select_main.obj`. Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o list_select list_select.c list_select_main.obj
```

Esempio 1

Lista dati in input:

```
21 -> -55 -> 69 -> -14 -> NULL
```

Lista posizioni in input:

```
3 -> 0 -> NULL
```

Lista in output:

```
-14 -> 21 -> NULL
```

Esempio 2

Lista in input:

```
2 -> 2 -> 0 -> NULL
```

Lista posizioni in input:

```
1 -> 1 -> 2 -> NULL
```

Lista in output:

```
2 -> 2 -> 0 -> NULL
```

2 Rimozione carattere (`remove_char.c`)

Scrivere una funzione **ricorsiva** C che rimuova tutte le occorrenze di uno specifico carattere da una stringa data in input, mentre questa viene copiata su una stringa in output.
La funziona dovrà essere case sensitive: maiuscole e minuscole sono caratteri distinti.
Lo spazio per la stringa in output è già stato allocato ed è suffiente a contenere l'intera stringa in input (nel caso in cui il carattere da rimuovere non sia presente).
La funzione da sviluppare è la seguente:

```
// Copia la stringa input in output, rimuovendo prima il carattere c
// La funzione deve essere case sensitive e implementata in maniera ricorsiva
void remove_char(char* input, char c, char* output);
```

Il programma di test è già implementato e compilato come file oggetto `remove_char_main.obj`. Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o remove_char remove_char.c remove_char_main.obj
```

Esempio 1

Stringa in input:

```
Ho attraversato gli oceani del tempo per trovarti
```

Carattere da rimuovere:

```
o
```

Stringa in output:

```
H attraversat gli ceani del temp per trvarti
```

Esempio 2

Stringa in input:

```
Fatti non foste a viver come BRUTI ma per seguir virtute e canoscenza
```

Carattere da rimuovere:

```
I
```

Stringa in output:

```
Fatti non foste a viver come BRUT ma per seguir virtute e canoscenza
```