Fondamenti di Informatica
Assegnamento 7

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 Double invert list (`double_invert.c`)

Scrivere una funzione C che data una lista concatenata, aggiunga in coda una lista di nuovi nodi in modo da raddoppiarne la lunghezza originale.
I nuovi nodi dovranno contenere gli stessi valori dei nodi originali, ma in ordine inverso. La funzione dovrà gestire il caso di lista vuota.

Il file header `double\_invert.h` contiene sia la struttura del nodo della lista che la firma della funzione da implementare:

```c
typedef struct Node
{
    int value;          // dato
    struct Node *next;  // link
} Node;

// Aggiunge in coda alla lista di input nuovi nodi in modo
// da raddoppiarne la lunghezza originale
// I nuovi nodi dovranno contenere gli stessi valori dei
// nodi originali, ma in ordine inverso
void double_invert(struct Node* head);
```

Utilizzare le funzioni viste in aula (provate a reimplementarle in autonomia, prima di guardare i sorgenti messi a disposizione) per creare nuovi nodi, inserirli in una lista e visualizzare le liste in fase di debug.
Una possibile soluzione al problema prevede di creare una nuova lista e inserire in ordine inverso gli elementi dalla lista in input, ad esempio utilizzando un inserimento in testa alla nuova lista.
Una volta riempita la nuova lista è sufficiente collegare il primo elemento della nuova lista all'ultimo elemento della lista originale per concludere l'operazione richiesta.
Il programma di test è già implementato e compilato, in forma di file oggetto `double_invert_main.obj`.
Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o double_invert double_invert.c double_invert_main.obj
```

Esempio 1

Lista in input:

```
2 -> -5 -> 6 -> -1 -> NULL
```

Lista in output:

```
2 -> -5 -> 6 -> -1 -> -1 -> 6 -> -5 -> 2 -> NULL
```

Esempio 2

Lista in input:

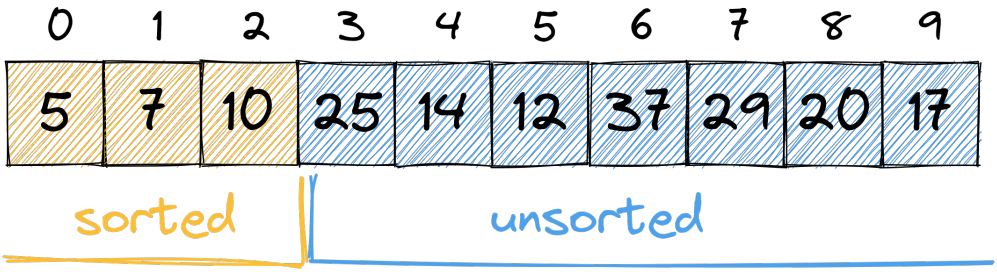
```
2 -> 2 -> 0 -> NULL
```

Lista in output:

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

2 Selection sort ricorsivo (`recursive_selection.c`)

Scrivere un programma C che implementi una versione ricorsiva di selection sort.
Vi ricordo che, selection sort ordina in-place dividendo logicamente gli elementi in due parti:



e spostando ad ogni iterazione il minimo valore di `unsorted` all'interno di `sorted`.
Nella versione iterativa, l'algoritmo è il seguente:

```c
void swap(int *a, int* b) {
    int temp = *a;
    *a = * b;
    *b = temp;
}

void selection_sort(int v[], int size) {
    for (int next=0; next < size-1; next++) {
        // Cerco il minimo a partire da next
        int min_index = next;
        for (int i = next+1; i < size; i++) {
            if (v[i] < v[min_index])
                min_index = i;
        }
        // Scambio next con il minimo trovato
        swap(&v[next], &v[min_index]);
    }
}
```

Dato `start`, puntatore al primo elemento dell'array da ordinare, e `end`, puntatore all'ultimo elemento dell'array da ordinare, si sviluppino le seguenti funzioni:

```c
// Trova ricorsivamente il minimo all'interno di un array da ordinare
int *recursive_min_index(int *start, int* end);

// Ordina ricorsivamente un array usando l'algoritmo selection sort
void recursive_selection_sort(int* start, int* end);
```

Dove `recursive_min_index()` restituisce il puntatore al minimo elemento tra `start` e `end`, mentre `recursive_selection_sort()` riordina ricorsivamente gli elementi tra `start` e `end` utilizzando al suo interno `recursive_min_index()` e `swap()`(invariato rispetto alla versione iterativa).

Esempio 1

Array in input:

```
[ 25 14 12 5 37 29 10 7 20 17 ]
```

Array in output:

```
[ 5 7 10 12 14 17 20 25 29 37 ]
```

Esempio 2

Array in input:

```
[ 5 5 10 5 5 ]
```

Array in output:

```
[ 5 5 5 5 10 ]
```