Fondamenti di Informatica
Assegnamento 5

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 Carriera studenti università (`univ.c`)

Scrivere un programma C che simuli l'archivio di un'università contenente la carriera degli studenti iscritti. Le informazioni memorizzate nell'archivio sono descritte all'interno del file header `univ.h` nelle seguenti strutture:

```c
// Lunghezza massima campi testuali
#define TEXT_LEN  20
// Numero massimo esami
#define MAX_EXAMS 30

// Struttura per la valutazione di un esame
typedef struct
{
    int mark;
    int credits;
} Evaluation;

// Struttura che rappresenta la carriera di uno studente
typedef struct
{
    char name[TEXT_LEN];
    char surname[TEXT_LEN];
    int  student_id;
    int  num_exams;
    Evaluation exams[MAX_EXAMS];
} Career;

// Stampa la carriera di uno studente, con o senza dettaglio sugli esami
void print_student(Career *student, bool with_exams);
```

La funzione `print_student` può essere usata in fase di sviluppo a scopo di debug, ad esempio prima e dopo la modifica di un campo della struttura, ma non deve essere richiamata in fase di autovalutazione e nell'elaborato da consegnare.
Qui sotto è riportato un esempio delle informazioni mostrate a video dalla funzione `print_student` per tutti gli studenti presenti nel sistema (che non cambiano mai). Per ogni studente, il numero, i crediti e i voti degli esami sono generati casualmente ad ogni iterazione.

```
>>>>>>>>>> Student <<<<<<<<<
Name: Petyr     Surname: Baelish        ID: 284728
Num exams: 8    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Bran      Surname: Stark          ID: 238100
Num exams: 1    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Tyrion    Surname: Lannister      ID: 211727
Num exams: 11   Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Robert    Surname: Baratheon      ID: 220810
Num exams: 19   Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Ramsay    Surname: Bolton         ID: 296855
Num exams: 1    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Theon     Surname: Greyjoy        ID: 217992
Num exams: 9    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Jon       Surname: Snow           ID: 237809
Num exams: 17   Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Samwell   Surname: Tarly          ID: 241111
Num exams: 24   Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Daenerys  Surname: Targaryen      ID: 219117
Num exams: 8    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Jaqen     Surname: Hghar          ID: 224103
```

```
Num exams: 13    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Jorah      Surname: Mormont        ID: 103746
Num exams: 17    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Davos      Surname: Seaworth       ID: 207476
Num exams: 12    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Catelyn    Surname: Tully          ID: 282413
Num exams: 13    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Olenna     Surname: Tyrell         ID: 203764
Num exams: 12    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Sandor     Surname: Clegane        ID: 206134
Num exams: 21    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Oberyn     Surname: Martell        ID: 204602
Num exams: 13    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Ilyn       Surname: Payne          ID: 288501
Num exams: 15    Weighted average:        0

>>>>>>>>>> Student <<<<<<<<<
Name: Beric      Surname: Dondarrion     ID: 269525
Num exams: 4     Weighted average:        0
```

L'archivio mantiene gli studenti all'interno di un array. Per rendere funzionale il sistema, è necessario implementare le seguenti funzioni:

```c
// Inserisce le informazioni reletive ad un esame nella carriera dello studente,
// restituisce il numero di esami nella carriera dello studente fino a raggiungere MAX_EXAMS
// se la soglia viene superata o se necessario gestire altri errori ritornare 0
int insert_exam(Career *student, Evaluation exam);

// Calcola la media ponderata degli esami svolti da uno studente
// se necessario gestire errori, ritornare 0.0f
float get_weighted_average(Career *student);

// Calcola una statistica su quanti studenti superano una determinata soglia in media ponderata
int statistics(Career *students, int num_students, float threshold);
```

Per il calcolo della media ponderata si utilizzi la seguente formula:

$$\text{Media ponderata} = \frac{\sum_{i=1}^{n} (x_i \cdot p_i)}{\sum_{i=1}^{n} p_i}$$

Attenzione: la consistenza dei puntatori è garantita, invece va gestito all'interno delle funzioni la possibilità che i puntatori assumano il valore `NULL`.
Il programma di test è già implementato e compilato, fornito in forma di file oggetto `univ_main.obj`. Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o univ univ.c univ_main.obj
```

Esempio 1

Input:

```
2
241111
30
6
241111
30
15
20
```

Output:

```
Exams to add:
Student ID: Mark: Credits:
Student ID: Mark: Credits:
Threshold:
////////////////////////////////////////
STUDENTS OVER THRESHOLD: 17
```

Esempio 2

Input:

```
0
24
```

Output:

```
Exams to add:
Threshold:
////////////////////////////////////////
STUDENTS OVER THRESHOLD: 11
```

2 Fake sort (`fake_sort.c`)

Scrivere un programma C che sfrutti i puntatori per ordinare un array composto da elementi contenenti nome, cognome, ed età di una persona all'interno di una struttura.
La grandezza e gli elementi dell'array sono predeterminati.
L'unica cosa su cui è possibile agire è l'inizializzazione di alcune variabili di supporto.
L'utente inserisce il tipo di ordinamento che vuole ottenere.
Ad ogni tipo di ordinamento è associato un carattere.
Le informazioni memorizzate nell'archivio sono descritte all'interno del file header `fake_sort.h` nelle seguenti strutture:

```c
// Lunghezza massima campi testuali
#define TEXT_LEN  20
// Numero persone nell'array
#define NUM_PEOPLE 10

// Tipi di ordinamento
typedef enum {
    NAME_ASC = 'N',
    NAME_DES = 'n',
    SURNAME_ASC = 'S',
    SURNAME_DES = 's',
    AGE_ASC = 'A',
    AGE_DES = 'a',
} SortType;

// Struttura che rappresenta una persona
typedef struct {
    char name[TEXT_LEN];
    char surname[TEXT_LEN];
    int  age;
} Person;
```

Gli elementi presenti nell'array iniziale sono descritti nella seguente tabella:

Name	Surname	Age
Natilia	Romanoff	26
Tony	Stark	34
Thor	Odinson	1527
Bruce	Banner	41
Peter	Parker	22
Scott	Lang	33
Wanda	Maximoff	35
James	Howlett	155
Clinton	Barton	38
Steven	Rogers	79

Preordianre la tabella e inizializzare correttamente gli elementi nelle variabili perché l'ordinamento risulti corretto per tutti i 6 possibili casi.

Esempio 1

Input:

```
N
```

Output:

```
Sort type:
Bruce, Banner, 41
Clinton, Barton, 38
James, Howlett, 155
Natilia,  Romanoff, 26
Peter, Parker, 22
Scott, Lang, 33
Steven, Rogers, 79
Thor, Odinson, 1527
Tony, Stark, 34
Wanda, Maximoff, 35
```

Esempio 2

Input:

```
a
```

Output:

```
Sort type:
Thor, Odinson, 1527
James, Howlett, 155
Steven, Rogers, 79
Bruce, Banner, 41
Clinton, Barton, 38
Wanda, Maximoff, 35
Tony, Stark, 34
Scott, Lang, 33
Natilia,  Romanoff, 26
Peter, Parker, 22
```