Parte di programmazione (usate i vostri fogli)

ESERCIZIO 4 (4 punti)

Implementare la funzione `void printDate(char *date)` che riceve come parametro una data nel formato "`dd-mm-aaaa`" e la stampa in versione estesa a schermo. Ad esempio, "`10-05-2002`" deve essere stampata come `10 Maggio 2002`. Se l'input non è formattato correttamente (ad esempio: "`09/12-1998`") e non è una data corretta (ad esempio: "`45-01-2001`" oppure "`10-13-2003`"), la funzione deve stampare un messaggio di errore. Per semplicità, assumere che tutti i mesi siano di 30 giorni.

ESERCIZIO 5 (6 punti)

Implementare la funzione `char* findSequence(char s[], char ch, int n)` che cerca una sequenza contenente `n` volte il carattere `ch` nella stringa `s`. Se la sequenza esiste, la funzione deve ritornare un puntatore alla prima occorrenza della sequenza. Altrimenti, deve ritornare `NULL`.

<u>Esempio:</u>

`findSequence("ciccioPasticcio",'c',2)` cerca la prima occorrenza della sequenza `cc` all'interno di Ci**c**cioPasticcio. Ritorna il puntatore al terzo carattere della stringa (in grassetto), che è il primo di tale sequenza.

`findSequence("ciccioPasticcio",'c',3)` ritorna `NULL`, perchè `ciccioPasticcio` non contiene la sequenza `ccc`.

ESERCIZIO 6 (8 punti)

In un programma C, uno studente dell'ultimo appello di Tecniche di Programmazione viene rappresentato con la seguente struttura:

```
typedef struct {char matricola[7]; char name[30]; int score;} student;
```

Un file di testo, il cui nome è passato come primo parametro da linea di commando, contiene i dati di al più 100 studenti, scritti senza un ordine specifico. Ogni linea contiene matricola (numero a 6 cifre senza spazi), nome e voto all'esame (intero tra 0 e 30).

Dato il seguente frammento di codice:

```
int main (int argc, char *argv[]) {
  student listStudents[100], *selStudents[100];
  int i, n = readStudents(listStudents, 100, argv[1]);
  int threshold = 18;
  int nSuff = selectStudents(listStudents, n, selStudents, threshold);
  printResults(selStudents,nSuff,argv[2]);
  … // rest of the program (omitted)
  return 0;
}
```

1) La funzione `readStudents` (non riportata) legge il file in input e salva i dati degli studenti che hanno preso parte all'esame nell'array `listStudents`; ritorna inoltre come risultato il numero di studenti letti.
2) La funzione `selectStudents` seleziona gli studenti che hanno ottenuto voto uguale o superiore alla soglia (nel frammento di codice, la soglia è 18). I puntatori agli studenti selezionati devono essere salvati nel vettore `selStudents`, mentre il numero totale di studenti selezionati deve essere ritornato al chiamante.
3) La funzione `printResults` stampa il risultato dell'esame in un file di output, il cui nome è passato come secondo argomento da linea di commando. La funzione stampa i voti degli studenti selezionati, uno studente per riga, in forma anomima secondo il formato: `<identification_code> <score>`, dove `identification_code` si ottiene aggiungendo 100 al numero di matricola originale. (Ad esempio: la matricola 123456 diventa 123556).

Implementa le funzioni **`selectStudents`** e **`printResults`**. I prototipi delle funzioni devono essere compatibili con le chiamate a funzione contenute nel codice.