UNIVERSITÀ DEGLI STUDI DI PARMA

Dipartimento di Ingegneria e architettura
Corso di Laurea in Ingegneria Informatica elettronica e delle telecomunicazioni
a.a. 2020-2021

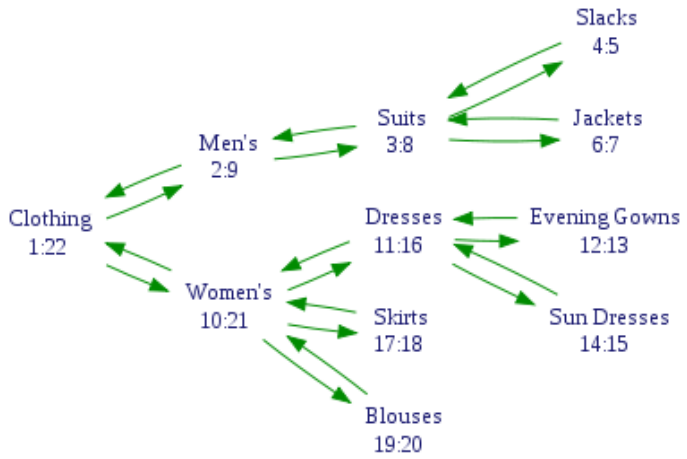
PROVA PRATICA DI INFORMATICA & LAB. PROGRAMMAZIONE
20 gennaio 2022

Nome: _____ Cognome:_____ Matr:_____ Postazione_____

Scrivere un programma in linguaggio C (chiamare il progetto con la propria **<matricola>**) che abbia il comportamento descritto nel seguito. Il tempo a disposizione e` di **120 minuti**. Al termine del tempo gli elaborati salvati su **Z:** verranno raccolti automaticamente dal sistema di laboratorio. Eventuali documenti sono disponibili in **P:\Bertozzi**, si consiglia di usare **wordpad** per leggere i file di testo.

In informatica capita spesso di dover gestire dati strutturati in maniera gerarchica. Ad esempio i possibili oggetti in vendita in un negozio online di abbigliamento potrebbero essere suddivisi in due categorie "uomo" e "donna". La categoria "uomo" potrebbe contenere altre sottocategorie ("pantaloni", "giubbotti"...) e ogni sottocategoria potrebbe conenerne altre ecc. ecc.
Tali strutture "ad albero" sono a volte gestite tramite i cosiddetti *Nested Set Model*.
In questo caso si associa a ciascun elemento della gerarchia due valori "lft" e "rgt" che permettono di gestire le relazioni tra i vari elementi della gerarchia. In particolare:

- Elementi allo stesso livello gerarchico (*siblings*) adiacenti sono tali che l'lft del secondo è uguale all'rgt del primo + 1
- Gli elementi (*sons*) gerarchicamente al di sotto di un altro elemento (*parent*) hanno valori rgt e lft racchiusi nell'intervallo determinato dai valori rgt e lft dell'elemento gerarchicamente superiore. In particolare il primo "figlio" ha lft uguale all'lft del "padre" + 1



Si consideri un file nested.csv che riga per riga contiene: un nome di categoria (massimo 200 caratteri), il valore lft e il valore rgt associati alla relativa categoria, in questo ordine e separati da ";"
Si sviluppi un programma in C che:

1. definisca opportuna struttura dati per memorizzare il contenuto di detto file ipotizzando che

non vi siano piú di 100 righe. Nel seguito tale struttura la indicheremo con “tree”.

2. legga e memorizzi il contenuto del file
3. contenga una funzione void print_nome(const char *s, const int level) che stampa la stringa passata preceduta da “level” spazi
4. contenga una funzione <indice> find_next_sibling(<tree>, <indice di partenza>) che prenda in ingresso la struttura dati e un’indice di partenza di uno degli elementi memorizzati, cerchi il primo elemento “successivo” allo stesso livello dell’elemento indicato e ne restituisca l’indice oppure un valore specifico qualora l’elemento passato non abbia “fratelli”
5. contenga una funzione <indice> find_first_son(<tree>, <indice di partenza>) che prenda in ingresso la struttura dati e un’indice di partenza di uno degli elementi memorizzati, cerchi il primo elemento “figlio” di quello passato ne restituisca l’indice oppure un valore specifico qualora non ne esistano
6. Contenga una funzione void print_hierarchy(<tree>, <indice di partenza>, const int level) che stampi una intera struttura gerarchica sfruttando la ricorsione. In modo schematico:

- La funzione, sfruttando la print_nome(), stampa l’elemento passato.
- Cerca se l’elemento passato abbia figli usando la find_first_son() e nel caso chiama ricorsivamente se stessa per stampare l’intera “sottogerarchia”
- Cerca i possibili elementi allo stesso livello e ancora una volta sfrutta la ricorsione per stampare le relative categorie e sottocategorie.
  Detta funzione viene invocata dalla main() con level=0.

Esempio di esecuzione:

```
ELECTRONICS
 TELEVISIONS
  TUBE
  LCD
  PLASMA
 PORTABLE ELECTRONICS
  MP3 PLAYERS
   FLASH
  CD PLAYERS
  2 WAY RADIOS
```