Esercizio 1

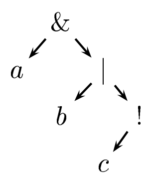
1. Definire in un file `assignment.cc` la struttura `assignment` come albero binario. Ogni nodo dell'albero deve contenere un carattere `var` e un booleano `value`: `var` rappresenta il nome di una variabile booleana, mentre `value` rappresenta il suo valore. L'albero deve essere ordinato rispetto al campo `var` e non deve contenere due nodi con lo stesso `var`. Definire le seguenti funzioni:

   - `assignment read_assignment ();`
     La funzione deve inizializzare un `assignment`, leggere da standard input una serie di coppie carattere/booleano e inserirle nell'`assignment`.
   - `void print_assignment (assignment);`
     La funzione deve stampare su standard output il contenuto dell'`assignment`.
   - `retval find (assignment t, char elem, bool &value);`
     La funzione deve trovare il nodo con `var==elem` e memorizzare in `value` il corrispondente valore booleano.

2. Definire in un file `formula.cc` la struttura `formula` come albero binario. Ogni nodo dell'albero deve contenere un carattere `symbol`. Ogni nodo di una `formula` correttamente costruita avrà la seguente forma:

   - (**foglie**) se `symbol`∈{`'0','1','a','b',..,'z'`} i figli del nodo sono vuoti;
   - (**not**) se `symbol=='!'` un figlio (per convenzione quello sinistro) contiene una formula non vuota mentre l'altro figlio è vuoto;
   - (**and/or**) se `symbol=='&'` o `symbol=='|'` entrambi i figli del nodo contengono una formula non vuota;

   E.g. la seguente struttura è corretta:



   Definire inoltre le seguenti funzioni:

   - `formula read_formula ();`
     La funzione deve leggere da standard input una formula in forma prefissa (e.g. `& a | b !c`) e creare la struttura corrispondente.
   - `void print_formula (formula);`
     La funzione deve stampare su standard output il contenuto della `formula` in forma infissa (e.g. `a & (b | !c)`).

   Suggerimento: per creare l'albero `formula` definire e utilizzare la funzione:

   - `formula new_formula (char symbol, formula left, formula right);`
     che presi in input due formule e un carattere, crea una terza formula corrispondente a `symbol`.

3. Aggiungere al file `formula.cc` la funzione
   `retval apply_assignment(formula &f, assignment a);`
   che deve scandire la struttura della formula `f` e sostiture alle occorrenze di ogni variabile il valore assegnato da `a` (se esiste). E.g. la formula
   `a & (b | !c)`
   con l'assegnamento {`b=0,a=1`} diventa
   `1 & (0 | !c)`.

4. Aggiungere al file `formula.cc` la funzione
   `retval simplify(formula &f);`
   che deve applicare ricorsivamente le seguenti regole di semplificazione:

   - !0 diventa 1
   - !1 diventa 0
   - α & 0 e 0 & α diventano 0
   - α & 1 e 1 & α diventano α
   - $\alpha \mid 0$ e $0 \mid \alpha$ diventano α
   - $\alpha \mid 1$ e $1 \mid \alpha$ diventano 1