University of Regina

Assignment #1

Prolog : The Unification Algorithm

Note: C|C++ and Java programs should compile|run on Hercules

Implement the unification algorithm seen in class. Your program should accept 2 terms at user input and returns the most general unifier of these 2 terms. Your program should have the same behavior as the unify_with_occurs_check(term1,term2) build-in predicate of Gnu Prolog (we do not consider however the case of lists). term1 and term2 are 2 terms and can be:

- ***Variables: starting with an upper case letter followed by any letter, digit or " _" (underscore)***
- ***Function symbols: starting with a lowercase letter followed by any letter, digit or " _" (underscore)***
- ***Other constants:***
  1. ***Numeric values: integer or real numbers.***
  2. ***Quoted items: any character(s) within a pair of single quotes ('Hello R2-D2 !')***

Examples:

|?- unify_with_occurs_check(a,X).

X = a

| ?- unify_with_occurs_check(X,Y12_).

Y12_ = X

| ?- unify_with_occurs_check(X,f(Y)).

X = f(Y)

| ?- unify_with_occurs_check(X,likes_same(a,Y)).

X = likes_same(13.199999999999999,Y)

?- unify_with_occurs_check(C_3PO,'Hello R2-D2 !').

C_3PO = 'Hello R2-D2 !'

| ?- unify_with_occurs_check(f(f(X,Y),X),f(f(V,U),g(U,Y))).

V = g(U,U)
X = g(U,U)
Y = U

Hand_In

A) IF YOUR ARE SUBMITTING 1 single FILE :

1. Name the file containing the C|C++ code "**assign1.cpp**". At the top of this file add the following comments :

    - instructions on how to compile your program,
    - an example on how to execute the program,
    - and other comments describing the program.

2. Submit your assignment through URCOURSES.

B) IF YOU ARE SUBMITTING MORE THAN ONE FILE :

Submit the following files through URCOURSES:

- **README** (file explaining the compilation and execution of your program including the format of input and other details)
- headers (.h)
- implementations (.cc , .cpp...etc)
- the Makefile :
    - should be named "**makefile**"
    - the generated executable should be named : "assign1"

You can give any name to your source files. The marker will only have to execute "**make**" to compile your program and "**assign1**" to run it.

Marking Scheme: **total = 100%**

1. Readability (program style) : 10%
    - Program easy to read,
    - well commented, good structured (layout, indentation, whitespace, ...) and designed(following the top-down approach).
2. Compiling and execution process : 10%
    - program compiles w/o errors and warnings

    - robustness : execution w/o run time errors
3. Correctness : 80%
    - code produces correct results (output)
