University of Regina

Assignment #1

Prolog : The Unification Algorithm

Note: Your submission should compile|run on hercules

Using C|C++ (or any other language as long as it compiles|runs on hercules with the UNIX system), implement the unification algorithm seen in class. Your program should have the same behavior as the unify_with_occurs_check(term1,term2) build-in predicate of Gnu Prolog. term1 and term2 are 2 terms where :

- ***variables are in upper case letters.***
- ***Function symbols are in lower case letters.***

Examples :

|?- unify_with_occurs_check(a,X).

X = a

yes

| ?- unify_with_occurs_check(X,Y).

Y = X

yes

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

X = f(Y)

yes

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

X = h(a,Y)

yes

| ?- 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++ (or others) code " **assign1username.cpp**"(or any other extension). At the top of this file add the following comments :
    - Your first name, last name and ID number,
    - 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 using UR Courses.

B) IF YOU ARE SUBMITTING MORE THAN ONE FILE :

submit, through UR Courses, the following files in a zip file named " **assign1username.zip**"

- **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 " **assign1username**" to run it.

Marking Scheme: total = 100%

1. Readability (program style) : 20%
    - Program easy to read,
    - well commented, good structured (layout, indentation, whitespace, ...) and designed(following the top-down approach).
2. Compiling and execution process : 20%
    - program compiles w/o errors and warnings
    - robustness : execution w/o run time errors
3. Correctness : 60%
    - code produces correct results (output)
