CSc 600-01 (Section 1)

Homework 3 - Logic Programming in Prolog

prepared by Ilya Kopyl

CSc 600 Homework 3 - Logic Programming in Prolog

March 13, 2018

Homework is prepared by: Ilya Kopyl.
It is formatted in LaTeX, using TeXShop editor (under GNU GPL license).
Diagrams are created in LucidChart online editor (lucidchart.com).

1. Write a PROLOG program that investigates family relationships using lists. The facts should be organized as follows:

```
m([first_male_name, second_male_name, ..., last_male_name]).
f([first_female_name, second_female_name, ..., last_female_name]).
family( [father, mother, [child1, child2, ..., child_n]] ).
```

Write rules that define the following relationships:

```
male(X)
female(X)
father, mother, parent
siblings1, siblings2
brother1, brother2
sister1, sister2
cousins
uncle, aunt
grandchild, grandson, granddaughter
greatgrandparent
ancestor
```

For each of these rules show an example of its use.

The answer is listed on the pages TBD through TBD.

The code listing of maxlen function:

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

The code listing of main program:

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

The result of the program execution:

```
Standard output:
```

2. Write a PROLOG program that includes the following operations with lists:

```
membership testing (is an element member of a list?)
first element
last element
two adjacent elements
three adjacent elements
append list1 to list2 producing list3
delete element from a list
append element to a list
insert element in a list
compute the length of list
reverse a list
check whether a list is a palindrome
display a list
```

For each of these operations write your implementation of the operation and show an example of its use. If a predicate already exists (predefined in Prolog), modify its name (e.g. myappend or append1). Lists to be processed can be created by an auxiliary program, defined as facts, or entered from the keyboard.

The answer is listed on the pages TBD through TBD.

The code listing of the two-dimensional array that stores bit pattern of each BigInt digit. It is declared in the global space (outside of any function).

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

Main program, excluding the declaration of BIG_DIGITS array:

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

3. Array processing (elimination of three largest values) (one of many array reduction problems)

The array a(1..n) contains arbitrary integers. Write a function reduce(a, n) that reduces the array a(1..n) by eliminating from it all values that are equal to three largest different integers. For example, if a=(9, 1, 1, 6, 7, 1, 2, 3, 3, 5, 6, 6, 6, 6, 7, 9) then three largest different integers are 6, 7, 9, and after reduction the reduced array would be a=(1, 1, 1, 2, 3, 3, 5), n=7. The time complexity of the solution should be in O(n).

The answer is listed on the pages 11 through 13.

The code listing of the entire program for problem #3:

```prolog
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
```