**Λογικός Προγραμματισμός**

**ΠΜΣ “Πληροφορική”**

**Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιά**

**2018-2019**

**Θέμα 1. Γενεαλογικό Δένδρο (10/100)**

Μπορούμε να αναπαραστήσουμε τις σχέσεις γονέων παιδιών με τον παρακάτω τρόπο :

/\*-------------------------------------------\*/

/\* parents(Child, Father, Mother) \*/

/\*-------------------------------------------\*/

parents(kostas,nikos,maria).

male(kostas).

ή τον παρακάτω τρόπο :

/\*-------------------------------------------\*/

/\* parents(Child, Parent) \*/

/\*-------------------------------------------\*/

parents(kostas,nikos).

parents(kostas,maria).

male(nikos).

male(kostas).

female(maria)

Επιλέξτε όποιο τρόπο επιθυμείτε και αναπαραστήστε το γενεαλογικό δένδρο της οικογένειά σας. Ορίσατε τις σχέσεις :

siblings(X,Y) % X,Y αδέλφια

father(Father,Child) % Father είναι πατέρας του Child

mother(Mother,Child) % Mother είναι μητέρα του Child

grandfather(GFather,Child) % GFather είναι παππούς του Child

grandmother(GMother,Child) % GMother είναι γιαγιά του Child

all\_kids(Kids,Father) % όλα τα παιδιά του Father σε μια λίστα Kids

first\_cousins(X,Y) % X,Y πρώτα ξαδέλφια

second\_cousins(X,Y) % X,Y δεύτερα ξαδέλφια

uncle(Uncle,X) % Uncle είναι θείος του/της X

aunt(Aunt,X) % Aunt είναι θεία του/της X

**Θέμα 2. Μικρά προγράμματα με λίστες (10/100)**

Να υλοποιήσετε τα κατηγορήματα :

1. sumlist(List,S). % List = λίστα αριθμών, S = άθροισμά τους
2. multlist(List,M). % List = λίστα αριθμών, M = γινόμενό τους
3. listlength(List,L). % List = λίστα, L = μήκος λίστας
4. first\_element(List,F). % List = λίστα, F = πρώτο στοιχείο της λίστας
5. last\_element(List,L). % List = λίστα, L = Τελευταίο στοιχείο της λίστας
6. nth\_element(List,N,E). % Επιστρέφει στο E το N-οστό στοιχείο της λίστας List
7. element\_position(List,E,N). % Επιστρέφει στο N την θέση του στοιχείου E της λίστας List
8. my\_member(X,List). % X είναι στοιχείο της λίστας List
9. my\_select(X,List,Rest) % Χ είναι στοιχείο της λίστας List, Rest η υπόλοιπη λίστα
10. my\_append(L1,L2,R). % R είναι το αποτέλεσμα της ‘ένωσης’ των λιστών L1, L2

**Θέμα 3. Πράξεις σε χάρτη (10/100)**

Αναπαριστούμε ένα 2-διάστατο χάρτη με μια διπλή λίστα :

% X 1 2 3 4 5 Y

map1([

[0,0,0,0,0], %1

[1,0,0,1,1], %2

[1,1,0,1,1], %3

[1,1,1,1,1], %4

[1,1,1,1,1] %5

]).

Να ορίσετε τα παρακάτω κατηγορήματα :

xymapelement(Map,X,Y,E). % το Ε είναι το περιεχόμενο του χάρτη Map στη θέση X,Y

mapelementposition(Map,E,X,Y). % βρίσκει τις συντεταγμένες X,Y του E στο χάρτη Map

printlist(List) % τυπώνει οριζόντια τα στοιχεία της λιστας List

printmap(Map) % τυπώνει όμορφα τον χάρτη Map

map\_maxXY(Map,MaxX,MaxY) % επιστρέφει τα MaxX και MaxY του χάρτη Map

**Θέμα 4. Depth First Search σε χάρτη (40/100)**

Αναπτύξτε πρόγραμμα που πραγματοποιεί αναζήτηση σε βάθος σε ένα 2-διάστατο χάρτη και εμφανίζει το μονοπάτι που βρήκε. Σας δίδεται μέρος του κώδικα :

% X 1 2 3 4 5 6 7 8 Y

map1([

[0,0,1,1,0,0,0,0], %1

[0,0,0,1,1,0,0,0], %2

[0,1,0,0,1,1,0,1], %3

[0,1,1,0,1,0,0,1], %4

[0,0,0,0,0,1,0,1], %5

[1,0,1,0,0,0,0,1] %6

]).

start( (1,1) ).

finish( (5,1) ).

/\*---------------------------------------------------\*/

% path\_from\_to(Path)

pathfinder(S,F,P) :-

map1(M),

findpath(M,S,F,[S],P).

path\_from\_to(P) :-

start(S), finish(F), map1(M),

findpath(M,S,F,[S],P).

findpath(M,F,F,\_,[F]).

findpath(M,S,F,Visited,[S|P]) :-

move(S,Next),

not(member(Next,Visited)),

valid(M,Next),

findpath(M,Next,F,[Next|Visited],P).

**Θέμα 5. Α Έκπληξη (15/100)**

A. Τι κάνει το παρακάτω πρόγραμμα; (10/100)

Τρέξτε τα ερωτήματα :

?- nomismata1(355,X).

?- nomismata2(355,X).

Διαβάστε τον κώδικα. Τι κάνει το πρόγραμμα; Μπορείτε να εξηγήσετε πως το κάνει;

nomismata1(Poso,[A100,A50,A20,A10,A5,A2,A1]) :-

member(A100,[0,1,2,3]),

member(A50,[0,1,2]),

member(A20,[0,1,2,3,4,5,6,7,8]),

member(A10,[0,1,2,3,4,5,6,7,8,9,10,11,12]),

member(A5, [0,1,2,3]),

member(A2, [0,1,2,3,4,5,6]),

member(A1, [0,1,2,3,4,5]),

Poso is A100\*100+A50\*50+A20\*20+A10\*10+A5\*5+A2\*2+A1\*1.

nomismata2(Poso,[A100,A50,A20,A10,A5,A2,A1]) :-

member(A1, [0,1,2,3,4,5]),

member(A2, [0,1,2,3,4,5,6]),

member(A5, [0,1,2,3]),

member(A10,[0,1,2,3,4,5,6,7,8,9,10,11,12]),

member(A20,[0,1,2,3,4,5,6,7,8]),

member(A50,[0,1,2]),

member(A100,[0,1,2,3]),

Poso is A100\*100+A50\*50+A20\*20+A10\*10+A5\*5+A2\*2+A1\*1.

**B. . Έκπληξη (15/100)**

Τι κάνει το παρακάτω πρόγραμμα;

Τρέξτε τα ερωτήματα :

?- differentiate(x^4+2\*x^3+3\*x^2+x+sin(2\*x)+3\*cos(exp(2\*x), R).

Διαβάστε τον κώδικα. Τι κάνει το πρόγραμμα differentiate, diff, simpl, simplify ; Μπορείτε να εξηγήσετε πως το κάνει; Γιατί η τερματική συνθήκη στο simpl είναι μετά από τον αναδρομικό κανόνα ; Τρέξτε και άλλα παραδείγματα.

differentiate(F, SDF) :- diff(F, DF), simpl(DF, SDF).

diff(C, 0) :- integer(C). /\* 1 \*/

diff(x, 1). /\* 2 \*/

diff(F+G, DF+DG) :- diff(F, DF), diff(G, DG). /\* 3 \*/

diff(-F, -DF) :- diff(F, DF). /\* 4 \*/

diff(F-G, DF-DG) :- diff(F, DF), diff(G, DG). /\* 5 \*/

diff(C\*F, C\*DF) :- integer(C), !, diff(F, DF). /\* 6 \*/

diff(F\*G, F\*DG+G\*DF) :- diff(F, DF), diff(G, DG). /\* 7 \*/

diff(F/G, (G\*DF-F\*DG)/G^2) :- diff(F, DF), diff(G, DG). /\* 8 \*/

diff(F^C, C\*F^(C-1)\*DF) :- integer(C), diff(F, DF). /\* 9 \*/

diff(sin(F), cos(F)\*DF) :- diff(F, DF). /\* 10 \*/

diff(cos(F), -sin(F)\*DF) :- diff(F, DF). /\* 11 \*/

diff(log(F), (1/F)\*DF) :- diff(F, DF). /\* 12 \*/

diff(exp(F), exp(F)\*DF) :- diff(F, DF). /\* 13 \*/

simpl(F, SF2) :- simplify(F, SF1), !, simpl(SF1, SF2).

simpl(F, F).

simplify(A+B, C) :- integer(A), integer(B), !, C is A+B.

simplify(A-B, C) :- integer(A), integer(B), !, C is A-B.

simplify(A\*B, C) :- integer(A), integer(B), !, C is A\*B.

simplify(0+F, F) :- !.

simplify(F+0, F) :- !.

simplify(1\*F, F) :- !.

simplify(F\*1, F) :- !.

simplify(0\*\_, 0) :- !.

simplify(\_\*0, 0) :- !.

simplify(F^1, F) :- !.

simplify(-F, -SF) :- simplify(F, SF).

simplify(F+G, F+SG) :- simplify(G, SG).

simplify(F+G, SF+G) :- simplify(F, SF).

simplify(F-G, F-SG) :- simplify(G, SG).

simplify(F-G, SF-G) :- simplify(F, SF).

simplify(F\*G, F\*SG) :- simplify(G, SG).

simplify(F\*G, SF\*G) :- simplify(F, SF).

simplify(F/G, F/SG) :- simplify(G, SG).

simplify(F/G, SF/G) :- simplify(F, SF).

simplify(F^G, F^SG) :- simplify(G, SG).

simplify(F^G, SF^G) :- simplify(F, SF).

simplify(F\*(G\*H), (F\*G)\*H).