ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Περιεχόμενα

1. Ασκήσεις "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2001-02

1.1 Άσκηση 1

Το φίδι των ΤΥΠΑ έχει στο σώμα του ένα επαναλαμβανόμενο σχήμα (όχι, κατ' ανάγκη ακέραιο αριθμό φορών), το οποίο αποτελείται από σχέδια καθένα από τα οποία μπορεί να παρασταθεί από ένα χαρακτήρα. Επίσης, είναι γνωστό ότι το φίδι των ΤΥΠΑ αρέσκεται να ξεκουράζεται σε μία "φιδοειδή" διευθέτηση καταλαμβάνοντας το χώρο ενός ορθογωνίου παραλληλογράμμου δεδομένων διαστάσεων (σε μονάδα μέτρησης το μέγεθος του σχεδίου/χαρακτήρα που συνθέτει τα επαναλαμβανόμενα σχήματα). Γράψτε ένα κατηγόρημα `typa_snake/3`, το οποίο όταν καλείται σαν `typa_snake(Pattern, Width, Height)` να εκτυπώνει τη στάση ξεκούρασης του φιδιού των ΤΥΠΑ, με επαναλαμβανόμενο σχήμα τους χαρακτήρες της λίστας `Pattern` και πλάτος (αντίστοιχα, ύψος) του ορθογωνίου που προαναφέρθηκε το μήκος της λίστας `Width` (αντίστοιχα, `Height`). Το πρόγραμμά σας δεν πρέπει να χρησιμοποιήσει καθόλου αριθμητική, δηλαδή δεν θα περιλαμβάνει κανένα από τα σύμβολα `is`, `<`, `>`, `>=`, `=<`, `+`, `-`, `*`, αλλά ούτε και τα `name`, `=..`, `arg`, `functor`. Ένα παράδειγμα εκτέλεσης του προγράμματος είναι το εξής:

```
?- typa_snake([a,b,c,d,e,f,g],
              [_,_,_,_,_,_,_,_,_,_,_],
              [_,_,_,_,_,_]).
abcdefgabcd
agfedcbagfe
bcdefgabcde
bagfedcbagf
cdefgabcdef
cbagfedcbag
yes
```

1.2 Άσκηση 2

Να ορισθεί ένα κατηγόρημα `powers/3` το οποίο όταν καλείται δίνοντας στο πρώτο όρισμά του μία λίστα ακεραίων μεγαλύτερων του 1 και στο δεύτερο όρισμα ένα θετικό ακέραιο *N*, να επιστρέφει στο τρίτο όρισμά του τη λίστα που περιέχει τους *N* μικρότερους ακέραιους σε αύξουσα σειρά που είναι θετικές δυνάμεις ακεραίων του πρώτου ορίσματος. Παραδείγματα:

```
?- powers([3,5,4],17,L).

L = [3,4,5,9,16,25,27,64,81,125,243,256,625,729,1024,2187,

     3125]



?- powers([2,3,4,5,6,7,8,9,10],50,L).

L = [2,3,4,5,6,7,8,9,10,16,25,27,32,36,49,64,81,100,125,128,

     216,243,256,343,512,625,729,1000,1024,1296,2048,2187,

     2401,3125,4096,6561,7776,8192,10000,15625,16384,16807,

     19683,32768,46656,59049,65536,78125,100000,117649]



?- powers([2,9999999,9999999],20,L).

L = [2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,
```

```
     32768,65536,131072,262144,524288,1048576]


?- powers([4,2],6,L).

L = [2,4,8,16,32,64]
```

1.3 Άσκηση 3

Έστω ότι έχετε στη διάθεσή σας ένα μικροεπεξεργαστή που διαθέτει *N* καταχωρητές R_1, R_2, …, R_N οι οποίοι έχουν δακτυλιοειδή διευθέτηση. Αυτό σημαίνει ότι μπορεί κάποιος να μεταφέρει τα περιεχόμενα του καταχωρητή R_i στον καταχωρητή R_{i+1}, για $1 \leq i < N$, και του R_N στον R_1 με τις εντολές `move`(i), για $1 \leq i < N$, και `move`(N), αντίστοιχα. Επίσης, μπορεί να αντιμεταθέσει τα περιεχόμενα των καταχωρητών R_i και R_j με την εντολή `swap(i,j)`, όπου `i < j`. Έστω, τώρα, ότι σας δίνονται τα αρχικά περιεχόμενα των N καταχωρητών, καθώς και τα επιθυμητά τελικά περιεχόμενα. Το ζητούμενο είναι να βρεθεί η μικρότερη αλληλουχία από τις εντολές `move` και `swap` που πρέπει να εκτελεσθούν για να επιτευχθεί ο ζητούμενος μετασχηματισμός. Συγκεκριμένα, ορίστε το κατηγόρημα `codegen/3`, έτσι ώστε όταν αυτό καλείται με πρώτο όρισμα τη λίστα των αρχικών περιεχομένων των καταχωρητών και με δεύτερο όρισμα τη λίστα των τελικών περιεχομένων, να επιστρέφει στο τρίτο όρισμα τη λίστα των απαραίτητων (ελάχιστων) εντολών που απαιτούνται για το μετασχηματισμό. Σημειώστε ότι είναι δυνατόν στην αναπαράσταση των περιεχομένων των καταχωρητών να έχουμε, τόσο στην αρχική όσο και στην τελική κατάσταση, το σύμβολο `*`, που σημαίνει, για μεν την αρχική κατάσταση "δεν ξέρω τι περιέχεται στον καταχωρητή", για δε την τελική κατάσταση "δεν με ενδιαφέρει τι περιέχεται στον καταχωρητή". Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- codegen([a,b,c,d],[a,d,a,b],L).
L = [move(2),move(1),swap(3,4),swap(2,3)]

?- codegen([a,*,c],[c,a,*],L).
L = [move(1),move(3)]

?- codegen([a,b,c],[a,a,*],L).
L = [move(1)]
```

1.4 Άσκηση 4

Φανταστείτε ότι σε ένα δωμάτιο έχετε έναν αριθμό τηλεοράσεων, όπου κάθε μία είναι δυνατόν να συντονιστεί σε ένα από *N* διαθέσιμα κανάλια, έστω τα *1, 2, …, N*. Υποθέστε ότι αρχικά οι τηλεοράσεις είναι συντονισμένες σε αυθαίρετα κανάλια και ότι θέλετε να τις συντονίσετε όλες στο ίδιο κανάλι. Έχετε στη διάθεσή σας ένα τηλεχειριστήριο του οποίου η μοναδική λειτουργία είναι να μπορείτε να συντονίσετε οποιαδήποτε τηλεόραση στο επόμενο κανάλι από αυτό που είναι συντονισμένη (το επόμενο κανάλι από το *N* είναι το 1), αλλά δεν μπορείτε να κάνετε αυτήν την ενέργεια δύο συνεχόμενες φορές στην ίδια τηλεόραση, αν δεν μεσολαβήσει η αλλαγή καναλιού τουλάχιστον κάποιας άλλης. Ορίστε ένα κατηγόρημα `remote/3` το οποίο όταν καλείται με πρώτο όρισμα μία λίστα που περιέχει τα κανάλια στα οποία είναι συντονισμένες αρχικά οι τηλεοράσεις μας και με δεύτερο όρισμα το πλήθος των διαθέσιμων καναλιών, να επιστρέφει στο τρίτο όρισμα μία λίστα με τους αύξοντες αριθμούς των τηλεοράσεων στις οποίες όταν εφαρμοσθεί διαδοχικά η λειτουργία του τηλεχειριστηρίου, να έχουμε σαν αποτέλεσμα όλες οι τηλεοράσεις τελικά να είναι συντονισμένες στο ίδιο κανάλι. Μας είναι αδιάφορο ποιο θα είναι το τελικό κανάλι συντονισμού, αλλά μας ενδιαφέρει να κάνουμε τον ελάχιστο αριθμό αλλαγών σε κανάλια. Κάποια παραδείγματα εκτέλεσης θα μπορούσαν να ήταν τα εξής:

```
?- remote([3,1,3,3],5,L).

L = [1,2,3,2,4,2]
```

```
?- remote([2,2],3,L).

L = []


?- remote([2,5,1,6,4,2,6],8,L).

L = [1,2,1,3,1,3,1,3,5,6,5,6,3,6,3,6]
```

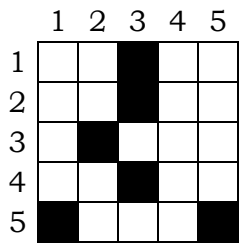
1.5 Άσκηση 5

Θεωρώντας τις συμβολοσειρές με στοιχεία μικρά γράμματα του λατινικού αλφαβήτου (από a έως z), θέλουμε να έχουμε μία μέθοδο συμπίεσης τέτοιων συμβολοσειρών. Η συμπίεση συνίσταται στην αναγνώριση επαναλαμβανόμενων μορφών στη συμβολοσειρά και στην αντικατάστασή τους με μία πιο συμπαγή μορφή. Για παράδειγμα, αν σε μία συμβολοσειρά έχουμε 7 επαναλήψεις του p, δηλαδή `ppppppp`, αυτό το τμήμα της συμβολοσειράς μπορεί να αντικατασταθεί από το `p7`. Αν η επαναλαμβανόμενη μορφή έχει μήκος μεγαλύτερο από ένα χαρακτήρα, πρέπει να χρησιμοποιήσουμε παρενθέσεις. Δηλαδή, το `ababab` πρέπει να γίνει `(ab)3`. Φυσικά, θα θέλαμε η συμπίεση να είναι όσο πιο πλήρης γίνεται. Δηλαδή, το `xcaabaabaabccadadcaabaabaabccadady` θα πρέπει να μετασχηματισθεί σε `x(c(a2b)3c2(ad)2)2y`. Γράψτε ένα κατηγόρημα `compress/2`, το οποίο να συμπιέζει τη συμβολοσειρά που του δίνουμε σαν λίστα στο πρώτο όρισμα και να επιστρέφει το αποτέλεσμα σαν λίστα στο δεύτερο όρισμα. Μερικά παραδείγματα εκτέλεσης είναι τα εξής:

```
?- compress([a,b,a,b,c,c,c,d,a,d,a,d],L).

L = ['(',a,b,')',2,c,3,'(',d,a,')',2,d]


?- compress([a,b,c,d],L).

L = [a,b,c,d]


?- compress([x,c,a,a,b,a,a,b,a,a,b,c,c,a,d,a,d,c,a,a,b,a,a,b,
             a,a,b,c,c,a,d,a,d,y],L).

L = [x,'(',c,'(',a,2,b,')',3,c,2,'(',a,d,')',2,')',2,y]
```

1.6 Άσκηση 6

Ένα σταυρόλεξο τετραγώνου σχήματος μπορεί να ορισθεί από τη διάστασή του και τις συντεταγμένες των μαύρων θέσεων. Για παράδειγμα, το σταυρόλεξο



μπορεί να περιγραφεί από τα γεγονότα Prolog:

```
dimension(5).

black(1,3).
black(2,3).
black(3,2).
black(4,3).
black(5,1).
black(5,5).
```

Δεδομένου και ενός γεγονότος με κατηγόρημα `words/1` μέσω του οποίου δίνονται οι λέξεις που πρέπει να τοποθετηθούν στο σταυρόλεξο, ορίστε ένα κατηγόρημα `crossword/1` που να επιστρέφει τη λίστα των διαθεσίμων λέξεων με τη σειρά που αυτές μπορούν να τοποθετηθούν στο σταυρόλεξο, πρώτα οι οριζόντιες και μετά οι κάθετες. Για παράδειγμα, αν δίνεται και το

```
words([do,ore,ma,lis,ur,as,po,so,pirus,oker,al,adam,ik]).
```

η σωστή απάντηση στο προηγούμενο σταυρόλεξο, δηλαδή αυτή που πρέπει να επιστρέψει το `crossword/1`, είναι η

```
[as,po,do,ik,ore,ma,ur,lis,adam,so,al,pirus,oker]
```

που αντιπροσωπεύει την εξής λύση:

	1	2	3	4	5
1	a	s	■	p	o
2	d	o	■	i	k
3	a	■	o	r	e
4	m	a	■	u	r
5	■	l	i	s	■

Αν τυχόν υπάρχουν περισσότερες της μίας λύσεις, πρέπει το `crossword/1` να τις υπολογίζει όλες μέσω οπισθοδρόμησης. Φυσικά, αν δεν υπάρχει λύση, πρέπει να αποτυγχάνει. Θεωρείται γνωστό, επίσης, ότι στα σταυρόλεξα δεν λογίζονται σαν λέξεις αυτές του ενός γράμματος. Δοκιμάστε το πρόγραμμά σας και σε μεγαλύτερα σταυρόλεξα, όπως:

```
dimension(20).

black(1,1).   black(1,3).   black(1,5).   black(1,7).
black(1,9).   black(1,16).  black(1,20).  black(2,10).
black(2,12).  black(2,14).  black(3,1).   black(3,3).
black(3,5).   black(3,7).   black(3,9).   black(3,19).
black(4,10).  black(4,12).  black(5,1).   black(5,3).
black(5,5).   black(5,16).  black(6,6).   black(6,8).
black(6,10).  black(6,11).  black(6,12).  black(6,18).
black(7,1).   black(7,3).   black(7,5).   black(7,12).
black(7,14).  black(7,15).  black(7,16).  black(7,17).
black(7,18).  black(7,20).  black(8,4).   black(8,9).
black(8,11).  black(8,20).  black(9,2).   black(9,7).
black(9,13).  black(9,15).  black(9,19).  black(10,2).
black(10,4).  black(10,6).  black(10,10). black(10,11).
black(10,19). black(11,8).  black(11,12). black(11,17).
black(12,2).  black(12,4).  black(12,6).  black(12,14).
black(12,16). black(13,6).  black(13,8).  black(13,14).
```

```
black(14,2).  black(14,4).  black(14,9).  black(14,15).
black(14,16). black(14,19). black(15,5).  black(15,6).
black(15,13). black(15,15). black(15,20). black(16,1).
black(16,5).  black(16,12). black(16,17). black(16,18).
black(16,20). black(17,6).  black(17,12). black(17,20).
black(18,2).  black(18,3).  black(18,4).  black(18,6).
black(18,7).  black(18,8).  black(18,11). black(18,17).
black(19,8).  black(19,14). black(19,15). black(20,2).
black(20,4).  black(20,6).  black(20,16).

words([ados,aere,aleser,alliee,apis,apivor,aria,arno,
       attire,attristee,ave,aviser,blocage,blonde,bol,ca,
       casse,caverne,client,colibri,cou,cultiver,dada,
       deite,demi,doter,eau,eclos,ecrase,ecrin,ecuri,
       eden,egri,elevation,emier,envol,eole,eolie,epte,
       ese,esope,et,etendard,etier,etoilee,etroitesse,
       europeenne,evier,evoe,feu,fraisier,gambader,gre,
       hesperides,id,ille,ils,in,ios,ir,isar,isole,lace,
       lave,le,les,levrette,lie,lieue,lippe,lo,mer,mulet,
       nato,nep,nie,noel,norme,notaire,ohe,ointe,ole,olt,
       orienter,oui,peler,pitre,puissante,rale,ratier,
       rose,savonner,soir,tavele,tirer,tresorier,trou,
       tulle,usa,usurper,utilise,va,valse,vilipe,voleter]).
```

1.7 Άσκηση 7

Να ορισθεί ένα κατηγόρημα `simplify/2` το οποίο να δέχεται στο πρώτο όρισμά του μία αλγεβρική έκφραση που μπορεί να περιλαμβάνει γινόμενα, αθροίσματα και διαφορές μεταξύ πολυωνύμων μίας μεταβλητής (έστω x) και να μετασχηματίζει αυτήν την έκφραση στο απλούστερο δυνατό πολυώνυμο που να το επιστρέφει στο δεύτερο όρισμα. Για παράδειγμα:

```
?- simplify((2*x^2-3*x+2)*(3*x-4)-(4*x^2+18*x-7),P).
P = 6*x^3-21*x^2-1
```

1.8 Άσκηση 8

Έστω ότι δίνεται η παρακάτω Prolog βάση δεδομένων:

```
borders(belgium,       france).
borders(belgium,       germany).
borders(belgium,       luxembourg).
borders(belgium,       netherlands).
borders(denmark,       germany).
borders(france,        belgium).
borders(france,        germany).
borders(france,        italy).
borders(france,        luxembourg).
borders(france,        spain).
borders(germany,       belgium).
borders(germany,       denmark).
borders(germany,       france).
borders(germany,       luxembourg).
borders(germany,       netherlands).
borders(great_britain, ireland).
borders(ireland,       great_britain).
borders(italy,         france).
borders(luxembourg,    belgium).
borders(luxembourg,    france).
borders(luxembourg,    germany).
```

```
borders(netherlands,   belgium).
borders(netherlands,   germany).
borders(portugal,      spain).
borders(spain,         france).
borders(spain,         portugal).

flows(achelous, greece).
flows(aliakmon, greece).
flows(danube,   germany).
flows(ebro,     spain).
flows(elbe,     germany).
flows(garonne,  france).
flows(garonne,  spain).
flows(loire,    france).
flows(maas,     belgium).
flows(maas,     france).
flows(maas,     netherlands).
flows(po,       italy).
flows(rhine,    france).
flows(rhine,    germany).
flows(rhine,    netherlands).
flows(rhone,    france).
flows(seine,    france).
flows(tagus,    portugal).
flows(tagus,    spain).
flows(thames,   great_britain).
flows(tiber,    italy).

belongs(brussels,   belgium).
belongs(copenhagen, denmark).
belongs(lyon,       france).
belongs(marseille,  france).
belongs(paris,      france).
belongs(berlin,     germany).
belongs(hamburg,    germany).
belongs(munich,     germany).
belongs(birmingham, great_britain).
belongs(glasgow,    great_britain).
belongs(london,     great_britain).
belongs(athens,     greece).
belongs(salonika,   greece).
belongs(dublin,     ireland).
belongs(milan,      italy).
belongs(naples,     italy).
belongs(rome,       italy).
belongs(luxembourg, luxembourg).
belongs(amsterdam,  netherlands).
belongs(hague,      netherlands).
belongs(rotterdam,  netherlands).
belongs(lisbon,     portugal).
belongs(madrid,     spain).
belongs(barcelona,  spain).
```

Τα κατηγορήματα `borders/2`, `flows/2` και `belongs/2` έχουν την προφανή σημασία. Ορίστε ένα κατηγόρημα `answer/2` που να είναι σε θέση να βρίσκει τις κατάλληλες απαντήσεις σε ερωτήσεις που είναι διατυπωμένες σε απλή φυσική Ελληνική γλώσσα[1]. Παραδείγματα:

```
?- answer('poies xwres synoreuoun me thn Ispania;',L).
L = [france,portugal]

?- answer('me poies xwres synoreuei h Gallia;',L).
L = [belgium,germany,italy,luxembourg,spain]

?- answer('poia potamia diarreoun thn Ellada;',L).
L = [achelous,aliakmon]

?- answer('poies xwres diarreei o Rhnos;',L).
L = [france,germany,netherlands]

?- answer('poies poleis briskontai sthn Ollandia;',L).
L = [amsterdam,hague,rotterdam]

?- answer('se poia xwra brisketai to Milano;',L).
L = [italy]

?- answer('me poies xwres tis opoies diarreei o Tagos
          synoreuei h Gallia;',L).
L = [spain]

?- answer('poies xwres oi opoies synoreuoun me to Louxembourgo
          synoreuoun me thn Italia;',L).
L = [france]

?- answer('poia xwra sthn opoia brisketai h Glaskwbh
          diarreei o Tameshs;',L).
L = [great_britain]

?- answer('poio potami to opoio diarreei to Belgio
          diarreei thn Ollandia;',L).
L = [maas]
```

1.9 Άσκηση[2] 9

Το πρόβλημα του χρωματισμού γράφου συνίσταται στην εύρεση του ελάχιστου αριθμού χρωμάτων που πρέπει να χρησιμοποιήσουμε για να χρωματίσουμε τους κόμβους δεδομένου γράφου, έτσι ώστε να μην υπάρχει ζευγάρι γειτονικών κόμβων που να έχουν το ίδιο χρώμα. Επιλύστε το πρόβλημα αυτό για γράφους με N κόμβους και πυκνότητα `D`, που θα κατασκευάζετε μέσω του `create_graph(N,D,G)` το οποίο θα βρείτε στο αρχείο `~takis/pr/graph.pl` (στο δίκτυο των Suns του τμήματος).

1.10 Άσκηση[3] 10

[1] Θα βοηθούσε ιδιαίτερα αν χρησιμοποιούσατε τη δυνατότητα που προσφέρουν διάφορα συστήματα Prolog (LPA Prolog, ECL^{i}PS^e, κλπ.) για την περιγραφή γλωσσών (φυσικών και άλλων) μέσω μίας γραμματικής οριστικών προτάσεων (definite clause grammar). Ανατρέξτε στο on-line help ή ζητήστε μου τα εγχειρίδια για περισσότερες λεπτομέρειες.

[2] Να αντιμετωπισθεί με τη βοήθεια της τεχνολογίας του λογικού προγραμματισμού με περιορισμούς στο περιβάλλον της γλώσσας ECLiPSe.

[3] Όπως και η άσκηση 9.

Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο Ν πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς `1, 2, 3, …, N`. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει Μ συνδυασμούς πτήσεων P_i $(1 \le i \le M)$. Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις `1, 2, 3, …, N` και, επίσης, τα P_i δεν είναι κατ' ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί που καλύπτουν ακριβώς τις πτήσεις της εταιρείας, με σκοπό να ανατεθούν σε συγκεκριμένους κυβερνήτες. Δηλαδή, δεν πρέπει ούτε να μείνει πτήση χωρίς κυβερνήτη, ούτε να κάποια πτήση να έχει δύο ή περισσότερους κυβερνήτες. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.) στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος. Εφαρμόστε τη μέθοδό σας για εισόδους που προκύπτουν από το `get_flight_data(I,N,P,C)` του προγράμματος `~takis/pr/flight_data.pl` (από το δίκτυο των Suns του τμήματος) δίνοντας στο `I` έναν αύξοντα αριθμό εισόδου `(1, 2, …)` και παίρνοντας στα `N, P` και `C` το πλήθος των πτήσεων, μία λίστα συνδυασμών πτήσεων και τη λίστα κοστών των συνδυασμών αυτών, αντίστοιχα.

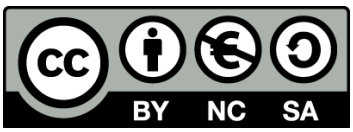
Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς

- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.