Θέματα προετοιμασίας (και λύσεις) για τις εξετάσεις στο μάθημα Αρχές Γλωσσών Προγραμματισμού
29/5/2025

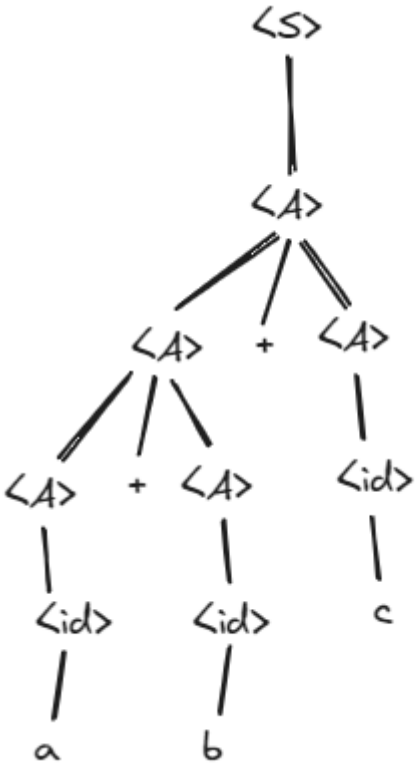
Ερώτημα 1

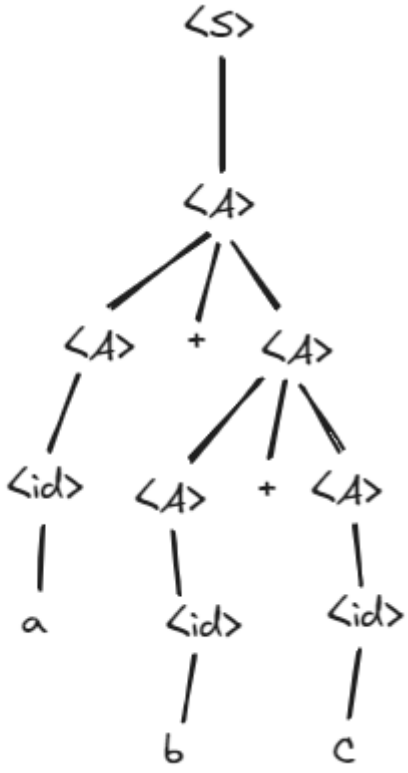
Δείξτε ότι η ακόλουθη γραμματική είναι ασαφής:

```
<S> -> <A>
<A> -> <A> + <A> | <id>
<id> -> a | b | c
```

Απάντηση:

Καθώς προκύπτουν διαφορετικά δένδρα συντακτικής ανάλυσης για την ίδια πρόταση, π.χ. για την πρόταση a+b+c, η γραμματική είναι ασαφής.





Ερώτημα 2

Δίνεται η ακόλουθη γραμματική:

```
<assign> -> <id> = <expr>
<id> -> X | Y | Z
<expr> -> <expr> + <term>
        | <term>
<term> -> <term> * <factor>
        | <factor>
<factor> -> ( <expr> )
          | <id>
```

Γράψε μια παραγωγή της πρότασης:

```
X = (X + Y * Z) * Z
```

Απάντηση: Μια παραγωγή (αριστερότερη) της πρότασης είναι η:

```
<assign> ==>
<id> = <expr> ==>
X = <expr> ==>
X = <term> ==>
X = <term> * <factor> ==>
X = <factor> * <factor> ==>
X = (<expr>) * <factor> ==>
```

```
X = (<expr> + <term>) * <factor> ==>
X = (<term> + <term>) * <factor> ==>
X = (<factor> + <term>) * <factor> ==>
X = (<id> + <term>) * <factor> ==>
X = (X + <term>) * <factor> ==>
X = (X + <term> * <factor>) * <factor> ==>
X = (X + <factor> * <factor>) * <factor> ==>
X = (X + <id> * <factor>) * <factor> ==>
X = (X + Y * <factor>) * <factor> ==>
X = (X + Y * <id>) * <factor> ==>
X = (X + Y * Z) * <factor> ==>
X = (X + Y * Z) * <id> ==>
X = (X + Y * Z) * Z
```

Ερώτημα 3

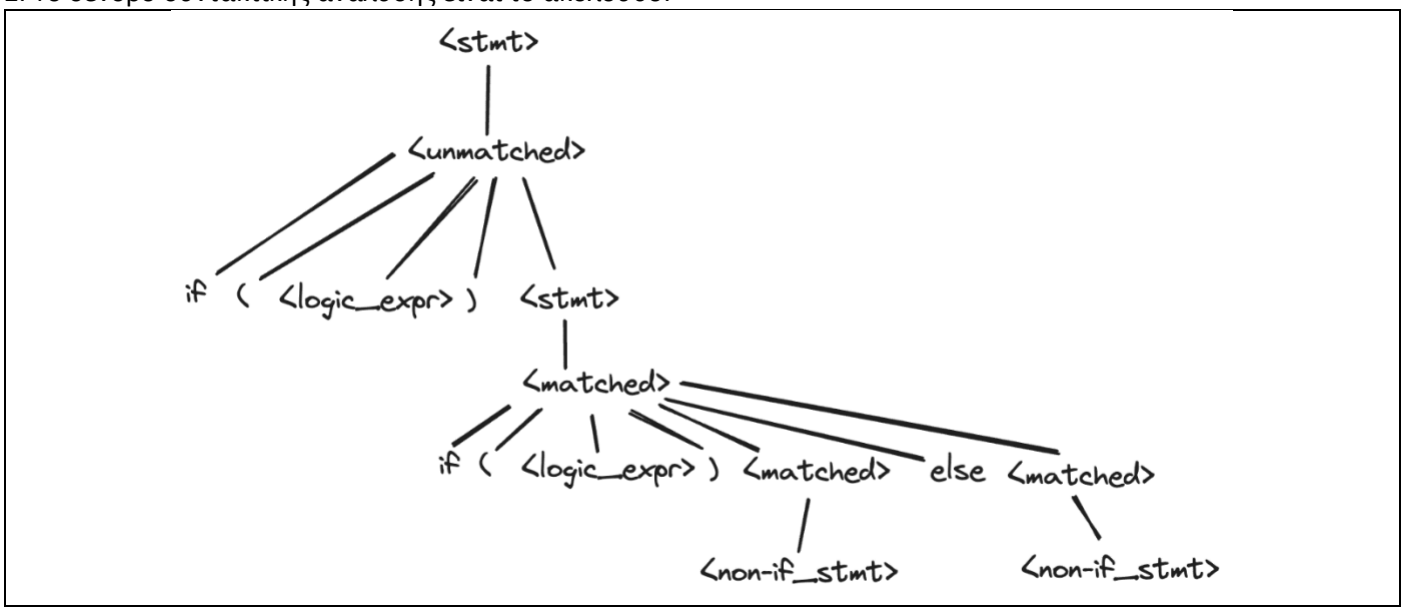
Δίνεται η ακόλουθη γραμματική:

```
<stmt> -> <matched> | <unmatched> | <non_if_stmt>
<matched>   -> if (<logic_expr>) <matched> else <matched>
            | <non-if_stmt>
<unmatched> -> if (<logic_expr>) <stmt>
            | if (<logic_expr>) <matched> else <unmatched>
```

1. Τι σημαίνει ότι η γραμματική αυτή επιλύει το πρόβλημα του dangling else;
2. Σχεδιάστε το δένδρο συντακτικής ανάλυσης για την πρόταση:
   `if (<logic_expr>) if (<logic_expr>) <non_if_stmt> else <non_if_stmt>`

Απάντηση:

1. Η γραμματική αυτή επιλύει το πρόβλημα του ξεκρέμαστου else, δηλαδή επιβάλει σε κάθε else να αφορά το πλησιέστερο προηγούμενο if. Αυτό γίνεται με τη διαφοροποίηση ανάμεσα σε matched και unmatched εντολές if.
2. Το δένδρο συντακτικής ανάλυσης είναι το ακόλουθο:



Ερώτημα 4

Δώστε ένα παράδειγμα κώδικα σε C στο οποίο να παραβιάζεται το χαρακτηριστικό των γλωσσών προγραμματισμού ορθογωνικότητα και σχολιάστε το.

Απάντηση:

```
#include <stdio.h>
#include <stdlib.h>

// Δυναμική κατανομή μνήμης για πίνακα 10 ακεραίων
void example1() {
```

```
  int *array = (int *)malloc(
      10 * sizeof(int)); // Χρησιμοποιεί τη malloc για να κατανομή μνήμης
  if (array == NULL) {
    printf("Memory allocation failed\n");
    return;
  }
  // Χρήση του array
  free(array); // Απελευθέρωση μνήμης
}

// Δυναμική κατανομή μνήμης για δισδιάστατο πίνακα ακεραίων 5 x 5
void example2() {
  int **matrix = (int **)malloc(
      5 * sizeof(int *)); // Χρησιμοποιεί τη malloc για τον πρώτο δείκτη
  if (matrix == NULL) {
    printf("Memory allocation failed\n");
    return;
  }
  for (int i = 0; i < 5; i++) {
    matrix[i] = (int *)malloc(
        5 * sizeof(int)); // Χρησιμοποιεί τη malloc για κάθε γραμμή
    if (matrix[i] == NULL) {
      printf("Memory allocation failed\n");
      return;
    }
  }
  // Χρήση του matrix
  for (int i = 0; i < 5; i++) {
    free(matrix[i]); // Απελευθέρωση μνήμης για κάθε γραμμή
  }
  free(matrix); // Απελευθέρωση μνήμης για τον πρώτο δείκτη
}

int main(void) {
  example1();
  example2();
  return 0;
}
```

Αν μια γλώσσα προγραμματισμού απαιτεί διαφορετικές συντακτικές μορφές ή μεθόδους για παρόμοιες λειτουργίες τότε παραβιάζει την ορθογωνικότητα και συνήθως ο κώδικας καθίσταται δυσκολότερος στην κατανόηση και συντήρηση. Στο παραπάνω παράδειγμα στη γλώσσα προγραμματισμού C η δέσμευση μνήμης για έναν δισδιάστατο πίνακα γίνεται διαφορετικά από ότι για έναν μονοδιάστατο πίνακα. Στον δισδιάστατο πίνακα απαιτούνται δύο επιπέδα κλήσης της malloc γεγονός που καθιστά τον κώδικα σχετικά πολύπλοκο.

Ερώτημα 5

Έστω ο ακόλουθος κώδικας σε μια υποθετική γλώσσα προγραμματισμού:

```
function outer(){
    function inner1() {
        var x = 5;
        inner2();
    }
    function inner2() {
        var y = x;
        print(y)
    }
    var x = 16;
    inner1();
}
outer()
```

Τι θα εμφανιστεί κατά την εκτέλεση αν η γλώσσα προγραμματισμού διαθέτει α) στατική εμβέλεια, β) δυναμική εμβέλεια;

Απάντηση:
Με στατική εμβέλεια εκτυπώνει 16, με δυναμική εμβέλεια εκτυπώνει 5

Ερώτημα 6
Δίνεται ο ακόλουθος κώδικας σε Python.

```
choice = input("1 για ακέραιους, 2 για πραγματικούς: ")
if choice == "1":
  print("Δώσε 2 ακεραίους: ", end="")
  a, b = input().split()
  print("Αποτέλεσμα: ", int(a) + int(b))
elif choice == "2":
  print("Δώσε 2 πραγματικούς: ", end="")
  a, b = input().split()
  print("Αποτέλεσμα: ", float(a) + float(b))
else:
  print("Λάθος επιλογή.")
```

Γράψτε κώδικα σε C που να έχει ισοδύναμη συμπεριφορά με τον παραπάνω κώδικα. Σε ποια περίπτωση από τους δύο κώδικες έχουμε δυναμική πρόσδεση τύπων και σε ποια στατική πρόσδεση τύπων;

Απάντηση:
Ο ισοδύναμος κώδικας σε C είναι ο ακόλουθος:

```
#include <stdio.h>

int main(void) {
  int choice;
  printf("1 για ακέραιους, 2 για πραγματικούς: ");
  scanf("%d", &choice);

  if (choice == 1) {
    int a, b;
    printf("Δώσε δύο ακεραίους: ");
    scanf("%d %d", &a, &b);
    printf("Αποτέλεσμα: %d\n", a + b);
  } else if (choice == 2) {
    float a, b;
    printf("Δώσε δύο πραγματικούς: ");
    scanf("%f %f", &a, &b);
    printf("Αποτέλεσμα: %.2f\n", a + b);
  } else {
    printf("Λάθος επιλογή\n");
  }

  return 0;
}
```

Στην περίπτωση της Python έχουμε δυναμική πρόσδεση τύπων (κατά το χρόνο εκτέλεσης), ενώ στη C έχουμε στατική πρόσδεση τύπων (κατά την μεταγλώττιση).

Ερώτημα 7
Δίνεται ο ακόλουθος κώδικας σε C. Τι θα συμβεί κατά τη μεταγλώττιση και σύνδεση; Τι σημαίνει ότι η μεταβλητή x είναι extern;

```c
#include <stdio.h>

int main() {
  extern int x;
  printf("%d", x);
}
```

Απάντηση:
Η μεταγλώττιση θα προχωρήσει κανονικά, αλλά η σύνδεση θα εμφανίσει μήνυμα σφάλματος καθώς δεν θα μπορεί να εντοπίσει τη μεταβλητή x που έχει οριστεί ως εξωτερική. Θα έπρεπε η μεταβλητή να είχε οριστεί σε ένα άλλο αρχείο πηγαίου κώδικα που να είχε συμπεριληφθεί στην εντολή μεταγλώττισης και σύνδεσης. Εναλλακτικά, η μεταβλητή x θα μπορούσε να είχε οριστεί μετά την main.

Ερώτημα 8
Τι θα εμφανίσει ο ακόλουθος κώδικας (σε C++) κατά την εκτέλεσή του;

```c
#include <iostream>

void foo() {
  static int x = 0;
  x++;
  std::cout << x << std::endl;
}

int main() {
  for (int i = 0; i < 5; i++) {
    foo();
  }
}
```

Απάντηση:
Καθώς οι static μεταβλητές διατηρούν τις τιμές τους σε διαδοχικές κλήσεις της συνάρτησης στην οποία έχουν δηλωθεί οι τιμές που θα εμφανιστούν κατά την εκτέλεση θα είναι:
1
2
3
4
5

Ερώτημα 9
Γράψτε μια συνάρτηση στη C που να δέχεται μια ακέραια παράμετρο με τιμή (call by value) και μια ακέραια παράμετρο με αναφορά (call by reference). Στο σώμα της συνάρτησης αυξήστε κατά ένα τις τιμές και των δύο παραμέτρων. Καλέστε τη συνάρτηση από τη main με ορίσματα τις τιμές δύο μεταβλητών με αρχικές τιμές μηδέν.

Απάντηση:

```c
#include <stdio.h>

void increment(int v, int* r) {
    v++;
    (*r)++;
}

int main() {
    int a = 0;
    int b = 0;

    increment(a, &b);

    printf("Value of a after increment: %d\n", a);
```

```
    printf("Value of b after increment: %d\n", b);

    return 0;
}
```

Ερώτημα 10

Δίνεται ο ακόλουθος κώδικας στη γλώσσα προγραμματισμού C. Συμπληρώστε τη συνάρτηση fun και την κλήση της στην main έτσι ώστε να εμφανίζει ο κώδικας την τιμή 2.

```
#include <stdio.h>

void fun(...) {
    ...
}

int main(void) {
  int x = 1;
  fun(...);
  printf("%d\n", x);
  return 0;
}
```

Απάντηση:

```
#include <stdio.h>

void fun(int *x) {
    *x = 2;
}

int main(void) {
  int x = 1;
  fun(&x);
  printf("%d\n", x);
  return 0;
}
```

Ερώτημα 11

1. Γράψτε στη C μια void συνάρτηση με όνομα fun που να δέχεται ως ορίσματα 3 ακεραίους και να επιστρέφει το δεύτερο όρισμα αυξημένο κατά την τιμή του πρώτου ορίσματος και το τρίτο όρισμα πολλαπλασιασμένο με την τιμή του πρώτου ορίσματος. Πραγματοποιήστε κλήση της συνάρτησης από την main για 3 μεταβλητές a, b, c με τιμές 5, 10 και 20 αντίστοιχα.
2. Επαναλάβατε το προηγούμενο ερώτημα, χρησιμοποιώντας τη γλώσσα C++, χρησιμοποιώντας αυτή τη φορά αναφορές (references) όπου χρειάζεται.

Απάντηση:

Για το 1)

```
#include <stdio.h>

void fun(int a, int* b, int* c) {
    *b += a;
    *c *= a;
}

int main(void) {
    int a = 5;
    int b = 10;
    int c = 20;
```

```
    fun(a, &b, &c);

    printf("Τιμή του b μετά την κλήση της fun: %d\n", b);
    printf("Τιμή του c μετά την κλήση της fun: %d\n", c);

    return 0;
}
```

Για το 2)

```
#include <iostream>

void fun(int a, int& b, int& c) {
    b += a;
    c *= a;
}

int main() {
    int a = 5;
    int b = 10;
    int c = 20;

    fun(a, b, c);

    std::cout << "Τιμή του b μετά την κλήση της fun: " << b << std::endl;
    std::cout << "Τιμή του c μετά την κλήση της fun: " << c << std::endl;

    return 0;
}
```

Ερώτημα 12

Τι θα εμφανίσει κατά την εκτέλεσή του ο ακόλουθος κώδικας;

```
#include <stdio.h>

void fun() {
  static int a = 0;
  extern int x;
  {
    int x = 0;
    x++;
    a++;
    printf("%d. x = %d\n", a, x);
  }
  a++;
  x++;
  printf("%d. x = %d\n", a, x);
}

int x = 5;

int main() {
  fun();
  fun();
}
```

Απάντηση:

Θα εμφανίσει:

```
1. x = 1
2. x = 6
3. x = 1
```

```
4. x = 7
```

Ερώτημα 13

Τι θα εμφανίσει κατά την εκτέλεση του ο ακόλουθος κώδικας C++; Ποια η σημασία του ορισμού της συνάρτησης fun της κλάσης A ως virtual;

```
#include <iostream>
using namespace std;
class A {
public:
  virtual void fun() { cout << "A" << endl; }
};
class B : public A {
public:
  void fun() { cout << "B" << endl; }
};
int main() {
  A *ref = new A();
  ref->fun();
  delete ref;
  ref = new B();
  ref->fun();
  delete ref;
  A obj = A();
  obj.fun();
  obj = B();
  obj.fun();
}
```

Απάντηση:

Θα εμφανίσει:

A

B

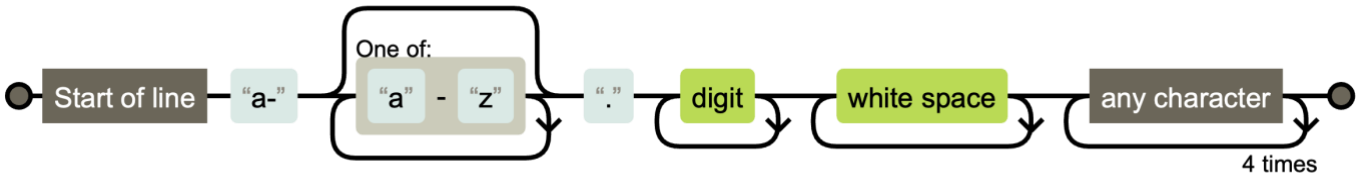
A

A

Εφόσον η συνάρτηση fun στην κλάση A έχει οριστεί ως virtual, αυτό σημαίνει ότι μπορεί να επαναοριστεί στις υποκλάσεις της A. Όταν καλείται η fun μέσω ενός δείκτη ή μιας αναφοράς σε αντικείμενο τύπου A, η πραγματική συνάρτηση που θα κληθεί θα είναι η συνάρτηση του αντικειμένου στο οποίο δείχνει ο δείκτης ή στο οποίο αναφέρεται η αναφορά που μπορεί να είναι είτε της κλάσης A είτε της κλάσης B.

Ερώτημα 14

Γράψτε την κανονική έκφραση που αντιστοιχεί στο ακόλουθο διάγραμμα:



Γράψτε 1 λεκτικό που θα εντόπιζε η συγκεκριμένη κανονική έκφραση.

Δίνεται ότι:

^ έναρξη γραμμής

. οποιοσδήποτε χαρακτήρας

\d ψηφίο

\s λευκός χαρακτήρας (π.χ. διάστημα, tab)

Απάντηση:

Η κανονική έκφραση που αντιστοιχεί στο διάγραμμα είναι: `^a-[a-z]*\.\d+\s+.{5}`
Ένα παράδειγμα λεκτικού που εντοπίζει η κανονική έκφραση είναι το:
`a-bcd.123  abcde`

Python

Ερώτημα 1

1. Γράψτε 2 διαφορετικούς τρόπους με τους οποίους μπορεί να πραγματοποιηθεί αντιμετάθεση δύο μεταβλητών x και y.
2. Κατασκευάστε μια λίστα a με 100 στοιχεία που να περιέχει εναλλάξ τις τιμές 0 και 1.
3. Δίνεται μια λίστα a με 1000 τιμές. Γράψτε κώδικα που να δημιουργεί μια νέα λίστα που να περιέχει μόνο τις διαφορετικές τιμές που υπάρχουν στη λίστα.
4. Δίνεται μια λίστα a με λεκτικά. Γράψτε κώδικα που να ταξινομεί σε αύξουσα σειρά τη λίστα με βάση το μήκος των λεκτικών.
5. Δίνονται οι ακόλουθες εντολές:

```
a = [1,2,3,4,5]
b = a
a[0], b[1] = 98, 99
print(a, b)
```

Τι θα εμφανιστεί κατά την εκτέλεση τους, γιατί;

Απάντηση:

```
1.
# α’ τρόπος αντιμετάθεσης των μεταβλητών x και y
x, y = y, x

# β’ τρόπος αντιμετάθεσης των μεταβλητών x και y
temp = x
x = y
y = temp

2.
# δημιουργία λίστας 100 θέσεων με 0 και 1, εναλλάξ
a = [i%2 for i in range(100)]

3.
# αφαίρεση διπλότυπων από λίστα σε νέα λίστα
b = list(set(a))

4.
# ταξινόμηση λίστας με λεκτικά με βάση το μήκος τους
a.sort(key=len)

5.
# θα εμφανίσει
[98,99,3,4,5], [98,99,3,4,5]
# διότι με την ανάθεση  b=a το a και το b είναι αναφορές προς τα ίδια δεδομένα.
```

Ερώτημα 2

Συμπληρώστε τη συνάρτηση hamming_distance έτσι ώστε να δέχεται δύο λεκτικά και να επιστρέφει την απόσταση Hamming (δηλαδή το πλήθος των αντίστοιχων χαρακτήρων που διαφέρουν στα δύο λεκτικά). Αν τα λεκτικά είναι διαφορετικού μήκους η συνάρτηση να επιστρέφει την τιμή -1. Προσθέστε 1 ακόμη περίπτωση ελέγχου της ορθής λειτουργίας της συνάρτησης hamming_distance.

```
import unittest

def hamming_distance(s, t):
    pass


class TestHammingDistance(unittest.TestCase):
```

```
    def test_HD(self):
        self.assertEqual(hamming_distance("GAGCCTACTAACGGGAT", "CATCGTAATGACGGCCT"), 7)

if __name__ == "__main__":
    unittest.main()
```

Απάντηση:

```
import unittest

def hamming_distance(s, t):
    if len(s) != len(t):
        return -1
    return sum([1 for i in range(len(s)) if s[i] != t[i]])

class TestHammingDistance(unittest.TestCase):
    def test_HD(self):
        self.assertEqual(hamming_distance("GAGCCTACTAACGGGAT", "CATCGTAATGACGGCCT"), 7)
        self.assertEqual(hamming_distance("GGT", "CATT"), -1)

if __name__ == "__main__":
    unittest.main()
```

Ερώτημα 3

Εξηγήστε τι συμπεριφορά του ακόλουθου κώδικα εντολή προς εντολή.

1	`import re`
2	`text = """One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly, slightly domed and divided by arches into stiff sections."""`
3	`results = re.findall(r"\b[a-z]+\b", text.lowercase())`
4	`results = set(results)`
5	`print(len(results))`

Απάντηση:

1. Εισαγωγή της εσωτερικής βιβλιοθήκης re
2. Λεκτικό πολλαπλών γραμμών στο οποίο θα γίνει η αναζήτηση για μοτίβα
3. Εύρεση όλων των λέξεων που αποτελούνται από πεζά γράμματα της αγγλικής αλφαβήτου
4. Μετατροπή της λίστας σε σύνολο για να αποφευχθούν οι διπλότυπες λέξεις
5. Εκτύπωση του πλήθους των μοναδικών λέξεων

Ερώτημα 4

Στην python ποια είναι η αναμενόμενη συμπεριφορά για τα ακόλουθα τμήματα κώδικα;

`a = [1, 2, 3, 4, 5]`
`for x in a:`
`    x += 1`
`print(a)`
	`a = [1, 2, 3, 4, 5]`
	`for i in range(len(a)):`
	`    a[i] += 1`
	`print(a)`
		`a = (1, 2, 3, 4, 5)`
		`for i in range(len(a)):`
		`    a[i] += 1`
		`print(a)`

(α)	(β)	(γ)

Απάντηση:

```
α)
[1, 2, 3, 4, 5]

β)
[2, 3, 4, 5, 6]

γ)
```

Θα εμφανιστεί σφάλμα (TypeError) καθώς το αντικείμενο a που είναι πλειάδα (tuple) δεν επιτρέπει αναθέσεις στα στοιχεία του

Ερώτημα 5

Τι θα εμφανίσουν οι ακόλουθοι κώδικες κατά την εκτέλεσή τους;

(α)	(β)

`g = 1`
 	
`def foo():`
`  g = 2`
`  print(g)`
 	
`print(g)`
`foo()`
`print(g)`
	`g = 1`
	
	`def foo():`
	`  g += 2`
	`  print(g)`
	
	`print(g)`
	`foo()`
	`print(g)`

Απάντηση:

α) Θα εκτυπώσει:
1
2
1

β) Θα εμφανίσει σφάλμα (UnboundLocalError) διότι εντός της συνάρτησης foo η πρώτη εντολή θεωρεί ότι αλλάζει μια τοπική μεταβλητή g χωρίς να της έχει ανατεθεί πρώτα τιμή

Ερώτημα 6

Γράψτε περιφραστικές λίστες (comprehensions) που:

1. Να δημιουργεί μια λίστα με όλα τα τετράγωνα των ακεραίων από το 1 μέχρι και το 10.
2. Να δημιουργεί μια λίστα με τους αντίστροφους αριθμούς όλων των ακεραίων αριθμών από το 10 μέχρι και το 20, στρογγυλοποιώντας στα 3 δεκαδικά ψηφία για κάθε αποτέλεσμα. Για τη στρογγυλοποίηση μπορείτε να χρησιμοποιήσετε τη συνάρτηση round, για παράδειγμα η κλήση round(5,6782) επιστρέφει 5.678.
3. Να δημιουργεί μια λίστα με τα μήκη των λεκτικών που βρίσκονται σε μια λίστα a_list. Σημείωση: η συνάρτηση len επιστρέφει το μήκος ενός λεκτικού που δέχεται ως όρισμα.

Απάντηση:

```
a = [i**2 for i in range(1, 11)]
print(a)

b = [round(1 / i, 3) for i in range(10, 21)]
print(b)

a_list = ["This", "is", "a", "list", "of", "strings"]
c = [len(s) for s in a_list]
print(c)

# [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# [0.1, 0.091, 0.083, 0.077, 0.071, 0.067, 0.062, 0.059, 0.056, 0.053]
# [4, 2, 1, 4, 2, 7]
```

Ερώτημα 7

Τι θα εμφανίσει ο ακόλουθος κώδικας:

```
text = "ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ"
s = set()
for c in text.split()[1]:
    s.add(c)
s = list(s)
s.sort()
```

```
print(s)
```

Απάντηση:

```
text = "ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ"
s = set()
for c in text.split()[1]: # διάσχιση των χαρακτήρων της δεύτερης λέξης του κειμένου
    s.add(c)
s = list(s)
s.sort() # ταξινόμηση των χαρακτήρων
print(s)
```

```
Θα εμφανίσει:
['Γ', 'Λ', 'Ν', 'Σ', 'Ω']
```

Ερώτημα 8

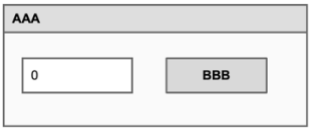
Περιγράψτε εν συντομία τον ρόλο των βιβλιοθηκών tkinter και matplotlib. Είναι ενσωματωμένες (builtin) βιβλιοθήκες της Python; Σχεδιάστε το τι θα εμφανίσει ο ακόλουθος κώδικας και περιγράψτε το ποια θα είναι η λειτουργικότητά του.

```
import tkinter as tk


def increment():
    x = int(count_label["text"])
    x += 1
    count_label["text"] = str(x)


root = tk.Tk()
root.geometry("200x30")
root.title("AAA")
count_label = tk.Label(root, text="0", width=10, background="white")
count_label.pack(side=tk.LEFT)
increment_button = tk.Button(root, text="BBB", width=10)
increment_button.pack(side=tk.RIGHT)
increment_button.config(command=increment)

root.mainloop()
```

Απάντηση:

Η βιβλιοθήκη tkinter είναι ενσωματωμένη βιβλιοθήκη στην Python και χρησιμοποιείται για τη δημιουργία Γραφικών Περιβαλλόντων Διεπαφής (GUIs).
Η βιβλιοθήκη matplotlib είναι εξωτερική βιβλιοθήκη και μπορεί να εγκατασταθεί με το pip, για παράδειγμα:
pip install matplotlib
Χρησιμοποιείται για σχεδιασμό γραφημάτων.

Θα εμφανίσει ένα παράθυρο με μια ετικέτα με αρχική τιμή 0 και ένα πλήκτρο, όπως στο ακόλουθο σχήμα.



Κάθε φορά που ο χρήστης θα πατά το πλήκτρο η τιμή στην ετικέτα θα αυξάνεται κατά 1.

Ερώτημα 9

1. Δίνεται η ακόλουθη κλήση λάμδα συνάρτησης (lambda x: (x + 3) * 5 / 2)(3). Τι πρόκειται να εμφανίσει;
2. Γράψτε μια λάμδα συνάρτηση που να δέχεται 2 παραμέτρους και να επιστρέφει το μέσο όρο τους. Καλέστε επαναληπτικά τη λάμδα συνάρτηση για κάθε πλειάδα της λίστας [(1,2), (2,3), (4,5)] και εμφανίστε το αποτέλεσμα κάθε κλήσης.
3. Γράψτε μια λάμδα συνάρτηση που να δέχεται 4 παραμέτρους a, b, wa, wb και να επιστρέφει το σταθμισμένο μέσο όρο των τιμών a και b με βάρη wa και wb αντίστοιχα (δηλαδή, αν οι παράμετροι λάβουν τις τιμές 5, 7, 0.3 και 0.7

αντίστοιχα να επιστρέφει $\frac{5*0.3+7*0.7}{0.3+0.7} = 6.4$). Στη συνέχεια, κατασκευάστε μια λίστα με 3 πλειάδες των 4 τιμών η κάθε μια, καλέστε επαναληπτικά τη λάμδα συνάρτηση για κάθε πλειάδα της λίστας και εμφανίστε με ένα δεκαδικό ψηφίο το αποτέλεσμα από την κάθε κλήση της λάμδα συνάρτησης.

Απάντηση:

```
1. Θα εμφανίσει: 15


2.
my_lambda = lambda x, y: (x + y / 2)

for x,y in [(1,2), (2,3), (4,5)]:
    print(my_lambda(x, y))

# 2.0
# 3.5
# 6.5


3.
my_lambda = lambda a, b, wa, wb: (a * wa + b * wb) / (wa + wb)

for a, b, wa, wb in [(1, 2, 0.5, 0.5), (3, 7, 0.8, 0.2), (9, 2, 0.6, 0.4)]:
    print(f"{my_lambda(a, b, wa, wb):.1f}")

# 1.5
# 3.8
# 6.2
```

Ερώτημα 10

Γράψτε μια συνάρτηση με όνομα reverse5 που να δέχεται ως παράμετρο ένα λεκτικό. Αν το πλήθος των χαρακτήρων του λεκτικού είναι μικρότερο από 5 να επιστρέφει None, αλλιώς να επιστρέφει τους 5 τελευταίους χαρακτήρες του λεκτικού σε ανεστραμμένη σειρά. Για παράδειγμα αν το λεκτικό είναι το «ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ» να επιστρέφει «ΥΟΜΣΙ».

Απάντηση:

```
def reverse5(str):
    if len(str) < 5:
        return None
    return str[-5:][::-1]

text = "ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ"
print(reverse5(text))
```

Ερώτημα 11

Τι θα εμφανίσει ο ακόλουθος κώδικας:

```
text = "ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ"
d = {}
for c in text:
    if c in d:
        d[c] += 1
    else:
        d[c] = 1

print([k for k in d if d[k] == 3])
```

Απάντηση:

Ο κώδικας δημιουργεί ένα λεξικό με κλειδιά του χαρακτήρες του κειμένου και τιμή για κάθε χαρακτήρα το πλήθος των χαρακτήρων. Τέλος, εμφανίζει τους χαρακτήρες που εμφανίζονται 3 φορές στο κείμενο. Δηλαδή θα εμφανίσει: ['A', 'P', 'M']

Ερώτημα 12

Γράψτε μια συνάρτηση calculate(a, b, operator) που λαμβάνει δύο αριθμούς a και b και έναν τελεστή ως λεκτικό ('+' ή '/'). Να διασφαλίζει ότι οι είσοδοι a και b είναι αριθμοί, αν όχι να προκαλείται ένα ValueError με κατάλληλο μήνυμα. Να εκτελεί την αριθμητική πράξη με βάση τον τελεστή που περνά, αν ο τελεστής δεν είναι κάποιος από τους '+' ή '/' τότε να προκαλείται ένα ValueError με κατάλληλο μήνυμα. Να χειρίζεται την περίπτωση διαίρεσης με μηδέν συλλαμβάνοντας την εξαίρεση ZeroDivisionError. Ειδικότερα να εμφανίζονται οι έξοδοι όπως φαίνεται στη συνέχεια:

```
print(calculate(10, 5, '+'))  # 15.0
print(calculate(10, 5, '/'))  # 2.0
print(calculate(10, 0, '/'))  # Σφάλμα: Διαίρεση με μηδέν.
print(calculate(10, 'a', '+'))      # Σφάλμα: Λάθος είσοδος. Εισάγετε μόνο αριθμούς.
print(calculate(10, 5, '*'))  # Σφάλμα: Λάθος τελεστής. Χρησιμοποιήστε μόνο '+' ή '/'.
```

Απάντηση:

```
def calculate(a, b, operator):
    try:
        # Μετατροπή εισόδων σε float και έλεγχος για ValueError
        a = float(a)
        b = float(b)
    except ValueError:
        return "Σφάλμα: Λάθος είσοδος. Εισάγετε μόνο αριθμούς."

    try:
        if operator == '+':
            return a + b
        elif operator == '/':
            return a / b
        else:
            raise ValueError("Λάθος τελεστής.")
    except ZeroDivisionError:
        return "Σφάλμα: Διαίρεση με μηδέν."
    except ValueError as e:
        return f"Σφάλμα: {e} Χρησιμοποιήστε μόνο '+' ή '/'."

# Έλεγχος της συνάρτησης calculate
print(calculate(10, 5, '+'))
print(calculate(10, 5, '/'))
print(calculate(10, 0, '/'))
print(calculate(10, 'a', '+'))
print(calculate(10, 5, '*'))
```

Haskell

Ερώτημα 1

Δίνεται η ακόλουθη συνάρτηση σε Haskell.

```
foo :: [Int] -> Int
foo [] = 1
foo (x:xs) = if x `mod` 2 == 0 then foo xs else x * foo xs
```

1. Τι σημαίνει η πρώτη γραμμή της συνάρτησης.
2. Τι θα επιστρέψει η κλήση της ως εξής: `foo [1,2,3,4,5,6,7]`

Απάντηση:

1.
Η γραμμή:

```
foo :: [Int] -> Int
```

σημαίνει ότι η συνάρτηση δέχεται ως όρισμα μια λίστα ακεραίων και επιστρέφει έναν ακέραιο.

2.
Η κλήση της συνάρτησης:

```
foo [1,2,3,4,5,6,7]
```

θα επιστρέψει το γινόμενο των περιττών όρων της λίστας, δηλαδή 1*3*5*7 που είναι ίσο με 105

Ερώτημα 2

Δίνεται ο ακόλουθος κώδικας:

```
foo :: Integer -> Integer
foo 0 = 1
foo x = x * foo (x - 1)

bar :: [Integer] -> Integer
bar [] = 1
bar (h : t) = h * bar t
```

1. Τι κάνει η συνάρτηση `foo` και τι η συνάρτηση `bar`;
2. Ποιο θα είναι το αποτέλεσμα για κάθε μια από τις ακόλουθες κλήσεις:

```
 I.   foo 4
II.   bar [1,2,3,4,5]
III.  foo (bar [1,2,3])
IV.   foo bar [1,2,3]
```

Απάντηση:

1.
Η συνάρτηση `foo` επιστρέφει το παραγοντικό ενός αριθμού. Η συνάρτηση `bar` επιστρέφει το γινόμενο των στοιχείων μιας λίστας.

2.

```
ghci> foo 4
24
ghci> bar [1,2,3,4,5]
120
ghci> foo (bar [1,2,3])
720
ghci> foo bar [1,2,3]
```

Επιστρέφει σφάλμα (error) διότι η συνάρτηση foo δέχεται 1 μόνο όρισμα ενώ της περνάμε 2 ορίσματα

Ερώτημα 3

Δίνεται ο ακόλουθος κώδικας:

```
foo :: (Ord a) => [a] -> [a]
foo [] = []
foo (x:xs) =
  let left = foo [a | a <- xs, a <= x]
      right = foo [a | a <- xs, a > x]
  in  left ++ [x] ++ right
```

1. Ποια είναι η λειτουργικότητα του κώδικα;

2. Αν ο κώδικας έχει τοποθετηθεί σε ένα αρχείο με όνομα haskell_erotima3.hs, πως μπορεί να φορτωθεί και να κληθεί μέσα από το ghci; Δώστε ένα παράδειγμα κλήσης της συνάρτησης `foo` και των αποτελεσμάτων που αναμένεται να ληφθούν.

Απάντηση:

1.
Η συνάρτηση `foo` δέχεται μια λίστα με στοιχεία που μπορούν να συγκρίνονται μεταξύ τους (μικρότερο, ίσο, κ.λπ.) και τα ταξινομεί σύμφωνα με τον αλγόριθμο της γρήγορης ταξινόμησης (quicksort).

2.

```
$ ghci
ghci> :load haskell_erotima3.hs
ghci> foo [4,5,2,3,1]
[1,2,3,4,5]
```

Ερώτημα 4

1. Υλοποιήστε τη συνάρτηση
`sumUpTo n`
που να υπολογίζει με αναδρομή το άθροισμα 1 + 2 + … + n.

2. Υλοποιήστε τη συνάρτηση
`power n k`
που να υπολογίζει με αναδρομή την ύψωση σε δύναμη n^k

Απάντηση:

```
sumUpTo :: Integer -> Integer
sumUpTo 1 = 1
sumUpTo n = n + sumUpTo (n - 1)

power :: Integer -> Integer -> Integer
power n 0 = 1
power n k = n * power n (k - 1)
```

Ερώτημα 5

Ποια θα είναι τα αποτελέσματα των ακόλουθων εντολών στο ghci;

```
1. ghci> (\x -> x * 2) 3
2. ghci> (\x y -> 2 * x + y) 1 2
3. ghci> map length ["arta", "ioannina", "preveza"]
4. ghci> map (*2) [1,2,3]
5. ghci> map (\x -> x+1) [1,2,3]
6. ghci> [1,3..10]
7. ghci> take 5 [0,5..]
8. ghci> filter even [1..10]
9. ghci> 9 `div` 4 + mod 10 4
```

10. ghci> (/) 5 2 + (*) 2 3

Απάντηση:

```
ghci> (\x -> x * 2) 3
6

ghci> (\x y -> 2 * x + y) 1 2
4

ghci> map length ["arta", "ioannina", "preveza"]
[4,8,7]

ghci> map (*2) [1,2,3]
[2,4,6]

ghci> map (\x -> x+1) [1,2,3]
[2,3,4]

ghci> [1,3..10]
[1,3,5,7,9]

ghci> take 5 [0,5..]
[0,5,10,15,20]

ghci> filter even [1..10]
[2,4,6,8,10]

ghci> 9 `div` 4 + mod 10 4
4

ghci> (/) 5 2 + (*) 2 3
8.5
```

Ερώτημα 6

Τι θα επιστρέψουν τα ακόλουθα comprehensions της Haskell;

```
1. ghci> [2*x|x <- [1..5]]
2. ghci> [x+y|x <- [0,5,10], y <- [1,2]]
3. ghci> [a^2|a <- [1..10], even a]
4. ghci> [reverse i | i <- ["arta", "ioannina", "preveza"]]
5. ghci> [i !! 1 | i <- ["arta", "ioannina", "preveza"]]
```

Απάντηση:

```
1.
ghci> [2*x|x <- [1..5]]
[2,4,6,8,10]

2.
ghci> [x+y|x <- [0,5,10], y <- [1,2]]
[1,2,6,7,11,12]

3.
ghci> [a^2|a <- [1..10], even a]
[4,16,36,64,100]

4.
ghci> [reverse i | i <- ["arta", "ioannina", "preveza"]]
["atra","aninnaoi","azeverp"]
```

```
5.
ghci> [i !! 1 | i <- ["arta", "ioannina", "preveza"]]
"ror"
```

Prolog

Ερώτημα 1

Έστω μια πύλη AND με 2 εισόδους.

1. Γράψτε το κατηγόρημα **and/3** με τα δύο πρώτα ορίσματα του (X και Y) να είναι οι είσοδοι και το τρίτο όρισμα (Z) να είναι η τιμή εξόδου (0 ή 1 αντίστοιχα).
2. Συνδυάζοντας 2 πύλες **and** κατασκευάστε ένα κύκλωμα που να δέχεται 3 εισόδους (X,Y,Z) και να επιστρέφει μια έξοδο (W) που θα είναι αληθής μόνο όταν και οι τρεις είσοδοι είναι αληθείς. Γράψτε το κατηγόρημα **and3/4** χρησιμοποιώντας το κατηγόρημα **and/3**.
3. Θέστε ερωτήματα που να επιστρέφουν για ποιες τιμές των εισόδων η έξοδος των πυλών είναι 0 τόσο για το κατηγόρημα **and/3** όσο και για το κατηγόρημα **and3/4**. Καταγράψτε και τα αποτελέσματα που επιστρέφουν τα ερωτήματα.

Απάντηση:

```
1.
and(0,0,0).
and(0,1,0).
and(1,0,0).
and(1,1,1).

2.
and3(X,Y,Z,W) :-
    and(X,Y,T),
    and(T,Z,W).

3.
?- and(X,Y,0).
X = 0, Y = 0 ;
X = 0, Y = 1 ;
X = 1, Y = 0.

?- and3(X,Y,Z,0).
X = 0, Y = 0, Z = 0 ;
X = 0, Y = 0, Z = 1 ;
X = 0, Y = 1, Z = 0 ;
X = 0, Y = 1, Z = 1 ;
X = 1, Y = 0, Z = 0 ;
X = 1, Y = 0, Z = 1 ;
X = 1, Y = 1, Z = 0.
```

Ερώτημα 2

Δίνεται η ακόλουθη βάση γνώσης:

```
parent(a,b).
parent(a,c).
parent(b,d).
parent(b,e).
parent(c,f).
descendent(X,Y) :- parent(Y,X).
descendent(X,Y) :- parent(Z,X), descendent(Z,Y).
```

Το κατηγόρημα **parent(X,Y)** ερμηνεύεται ως ότι ο **X** είναι γονέας του **Y**, ενώ το κατηγόρημα **descendent(X,Y)** ερμηνεύεται ως ότι ο **X** είναι απόγονος του **Y**.

1. Περιγράψτε λεκτικά το τι σημαίνουν οι δύο προτάσεις που συνιστούν το κατηγόρημα **descendent(X,Y)**.
2. Θέστε τα ακόλουθα ερωτήματα:
   I. Είναι ο **f** απόγονος του **b**;
   II. Ποιοι είναι οι απόγονοι του **a**;
   III. Ποιοι είναι οι πρόγονοι του **f**;

Απάντηση:

1.
Η πρώτη πρόταση σημαίνει ότι αν ο Y είναι γονέας του X τότε συνεπάγεται ότι ο X είναι απόγονος του Y.
Η δεύτερη πρόταση σημαίνει ότι αν ο Z είναι γονέας του X και ο Z είναι απόγονος του Y τότε συνεπάγεται ότι ο X είναι απόγονος του Y.

2.I
```
?- descendent(f, b).
false.
```

2.II
```
?- descendent(X, a).
```

2.III
```
?- descendent(f, X).
```

Ερώτημα 3
Ποια θα είναι τα αποτελέσματα των ακόλουθων ερωτημάτων;
```
1. ?- [H|T] = [1,2,3].
2. ?- [_,X,_|T] = [1,2,3,4,5].
3. ?- X = 1, Y is X + 1.
4. ?- member(X,[1,2,3]).
5. ?- member(X,[a,b]), member(Y,[1,2]).
6. ?- X = 1, Y = X + 1.
7. ?- X = 1 + 2, 1 + 2 = Y.
8. ?- X = 1, X = 2.
9. ?- X is Y + 2, Y = 2.
10.   ?- append(L1,L2,[1,2,3]).
```

Απάντηση:

```
1.
H = 1,
T = [2, 3].

2.
X = 2,
T = [4, 5].

3.
X = 1,
Y = 2

4.
member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3.



5.
X = a, Y = 1 ;
X = a, Y = 2 ;
X = b, Y = 1 ;
X = b, Y = 2.

6.
```

```
X = 1,
Y = 1+1.

7.
X = Y, Y = 1+2.

8.
false.

9.
ERROR: Arguments are not sufficiently instantiated

10.
L1 = [],
L2 = [1, 2, 3] ;
L1 = [1],
L2 = [2, 3] ;
L1 = [1, 2],
L2 = [3] ;
L1 = [1, 2, 3],
L2 = [] ;
false.
```

Ερώτημα 4

Δίνεται ο ακόλουθος ορισμός για το κατηγόρημα append/3:

```
append([], List, List).
append([Head | List_1], List_2, [Head | List_3]) :-
            append(List_1, List_2, List_3).
```

1. Περιγράψτε τη λειτουργία του κατηγορήματος append/3.
2. Δώστε 3 παραδείγματα όπου η χρήση του κατηγορήματος append/3 γίνεται με διαφορετικό τρόπο.
3. Εξηγήστε την υλοποίησή του append/3, αναλύοντας τις 2 προτάσεις από τις οποίες αποτελείται.

Απάντηση:

1.
Το κατηγόρημα append/3 χρησιμοποιείται για τη συνένωση 2 λιστών σε 1 τρίτη λίστα. Επίσης μπορεί να χρησιμοποιηθεί για τη διάσπαση μιας λίστας σε 2 μέρη ή για να ελεγχθεί αν μια λίστα είναι το τέλος μιας άλλης λίστας.

2.
Μερικά παραδείγματα χρήσης του append/3 είναι:
α) Συνένωση 2 λιστών σε 1 άλλη λίστα

```
?- append([1,2], [3,4], L).
L = [1,2,3,4].
```

β) Όλοι οι πιθανοί τρόποι διαχωρισμού μιας λίστας σε 2 επιμέρους λίστες

```
?- append(L1,L2, [a,b]).
L1 = [],
L2 = [a, b] ;
L1 = [a],
L2 = [b] ;
L1 = [a, b],
L2 = [] ;
false.
```

γ) Έλεγχος αν μια λίστα εφόσον προσαρτηθεί σε μια άλλη θα δώσει μια τρίτη λίστα.

```
?- L = [4,5], append([1,2,3],L,[1,2,3,4,5]).
```

```
L = [4, 5].
```

3.
Η πρώτη πρόταση (βασική περίπτωση της αναδρομής) σημαίνει ότι αν η πρώτη παράμετρος του κατηγορήματος append/3 είναι η κενή λίστα τότε η τρίτη παράμετρος θα είναι ίση με τη δεύτερη παράμετρο.

Η δεύτερη πρόταση (αναδρομική περίπτωση) σημαίνει ότι αν η πρώτη παράμετρος είναι μια λίστα με πρώτο στοιχείο το Head, τότε η τρίτη παράμετρος θα είναι μια λίστα με πρώτο στοιχείο το Head που θα ακολουθείται από μια λίστα (την List_3) που θα προκύπτει ως προσάρτηση της List_2 στη List_1.

Ερώτημα 5

Γράψτε το κατηγόρημα product_list/2 που θα επιστρέφει το γινόμενο των στοιχείων μιας λίστας. Για παράδειγμα:

```
?- product_list([1,2,3], X).
X = 6.
```

Απάντηση:

```
% Βασική περίπτωση: Το γινόμενο της άδειας λίστα είναι 1.
product_list([], 1).

% Αναδρομική περίπτωση: Το γινόμενο της λίστας είναι το γινόμενο του
% πρώτου στοιχείου με το γινόμενο της υπόλοιπης λίστας.
product_list([Head | Tail], Product) :-
    product_list(Tail, TailProduct),
    Product is Head * TailProduct.
```

Ερώτημα 6

Δίνεται η ακόλουθη βάση γνώσης σε Prolog:

```
woman(helen).
woman(maria).
woman(sofia).
loves(nikos, helen). % ο Νίκος αγαπά την Ελένη
loves(panos, helen).
loves(petros, viki).
loves(viki, petros).
jealous(X,Y):-
      loves(X,Z),
      loves(Y,Z).
```

1. Πόσες προτάσεις, πόσα γεγονότα και πόσους κανόνες περιέχει η βάση γνώσης;
2. Ποια κατηγορήματα, ποιους ατομικούς όρους και ποιες μεταβλητές περιέχει η βάση γνώσης;
3. Ερμηνεύστε λεκτικά την τελευταία πρόταση της βάσης γνώσης.
4. Αν η βάση γνώσης βρίσκεται σε ένα αρχείο με όνομα prolog_erotima6.pl, πως μπορεί να φορτωθεί στη γραμμή εντολών, από το swipl.
5. Ποια θα είναι τα αποτελέσματα εκτέλεσης των ακόλουθων ερωτημάτων;

```
 I.   ?- woman(X).
II.   ?- loves(panos,X), woman(X).
III.  ?- loves(petros,X), woman(X).
IV.   ?- jealous(panos,X).
```

Απάντηση:

1.
Η βάση γνώσης περιέχει 8 προτάσεις από τις οποίες οι 7 είναι γεγονότα και 1 είναι κανόνας.

2.
Τα κατηγορήματα είναι τα: woman/1, loves/2 και jealous/2
Οι ατομικοί όροι είναι οι: helen, maria, sofia, viki, petros

Οι μεταβλητές που αναφέρονται στη βάση γνώσης είναι οι: X, Y, Z

3.
Η τελευταία πρόταση της βάσης γνώσης είναι κανόνας και σημαίνει ότι αν ο X αγαπά τον Z και ο Y αγαπά τον Z τότε ο X ζηλεύει τον Y.

4.
Ένας τρόπος φόρτωσης της βάσης γνώσης, δεδομένου ότι βρισκόμαστε στον ίδιο φάκελο με το αρχείο prolog_erotima6.pl, είναι γράφοντας:

```
$ swipl -l prolog_erotima6.pl
```

Εναλλακτικά η φόρτωση της βάσης γνώσης μπορεί να γίνει ως εξής:

```
$ swipl
?- consult('prolog_erotima6.pl').
```

5.I

```
?- woman(X).
X = helen ;
X = maria ;
X = sofia.
```

5.II

```
?- loves(panos,X), woman(X).
X = helen.
```

5.III

```
?- loves(petros,X), woman(X).
false.
```

5.IV

```
?- jealous(panos,X).
X = nikos ;
X = panos.
```