Θεμα 3

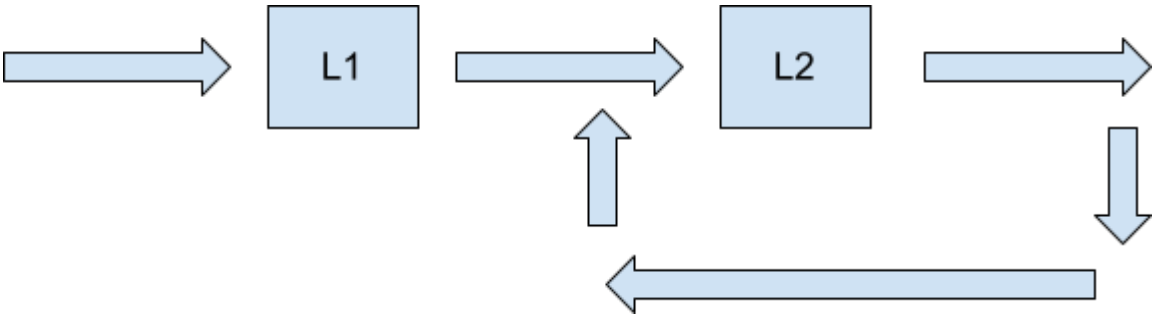
A) i)

Όνομα Φοιτητή 1: FOTIS **TSIOUMAS**
Α.Μ. Φοιτητή 1: ΜΠΠΛ**21079**
Όνομα Φοιτητή 2: **K**ONSTANTINOS **PETROU**
Α.Μ. Φοιτητή 2: ΜΠΠΛ**21062**
Alphabet: Σ= {**A, E, F, I, K, M, O, P, R, S, T, U, 0, 1, 2, 6, 7, 9**}

A) ii)

Regular Expression: L1L2*
οπου **L1** = {F, K, P, T} και **L2** = {A, E, I, M, O, P, R, S, T, U, 0, 1, 2, 6, 7, 9}

Συντακτικό Διάγραμμα:



B)

1. Λεκτική Ανάλυση

Κατα τη Λεκτικής Ανάλυσης διαβάζονται οι χαρακτήρες του source code και ομαδοποιούνται σε tokens/lexemes. Τα lexemes μπορεί να είναι δεσμευμένες λέξεις από τη γλώσσα (π.χ. if, for, while κτλ), operators (π.χ +, -, ==, =, ; κτλ), identifiers (π.χ. K, K2 κτλ) κ.α. Επίσης, υπάρχουν τα lexical values που έχουν εκχωρηθεί στους idendifiers. Επομένως, ο λεκτικός αναλυτής παίρνει ολόκληρο τον κώδικα του θέματος 3 σαν είσοδο και η πρώτη δουλειά που κάνει είναι να τον χωρίσει στα παρακάτω lexemes (όχι με την παρακάτω σειρά):
int, **if**, **iff**, **printf**, “**=**”, “**==**”, “**;**”, “**>**” , “**{**”, “**}**”, “**(**”, “**)**”, “**"**”, **K**, **K1**, **K2**, **K3**, **K4**, **K5**, **K6**, **K7**, **K8**, **K9**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **A**, **A1,** “**K is equal to K1**”, “**K is greater than K1**”

Αφού εισαχθούν αυτά τα lexemes και αναλυθούν, δημιουργείται ένας πίνακας συμβόλων από τις παρακάτω ομάδες (όχι με την παρακάτω σειρά):
Ομάδα 1: **int** Ομάδα 2: **if** Ομαδα 3: **iff** Ομάδα 4: **printf**

Ομάδα 5: **;**	Ομάδα 6: **=**	Ομάδα 7: **==**	Ομάδα 8: **>**
Ομάδα 9: **{**	Ομάδα 10: **}**	Ομάδα 11: **(**	Ομάδα 12: **)**
Ομάδα 13: **"**	Ομάδα 14: **id1**	Ομάδα 15: **id2**	Ομάδα 16: **id3**
Ομάδα 17: **id4**	Ομάδα 18: **id5**	Ομάδα 19: **id6**	Ομάδα 20: **id7**
Ομάδα 21: **id8**	Ομάδα 22: **id9**	Ομάδα 23: **id10**	Ομάδα 24: **id11**
Ομάδα 25: **2**	Ομάδα 26: **3**	Ομάδα 27: **4**	Ομάδα 28: **5**
Ομάδα 29: **6**	Ομάδα 30: **7**	Ομάδα 31: **8**	
Ομάδα 32: **K is equal to K1**		Ομάδα 33: **K is greater than K1**	

Τέλος ο λεκτικός αναλυτής παράγει το παρακάτω κώδικα και τον περνάει στον συντακτικό αναλυτή:

```
int id0 = 2;
int id1 = 3;
int id2 = 4;
int id3 = 5;
int id4 = 4;
int id5 = 5;
int id6 = 4;
int id7 = 6;
int id8 = 7;
int id9 = 8;
if id0==id1 {
   printf("K is equal to K1");
}
iff id10 > id11 {
   printf("K is greater than K1");
}
id1 = id4;
id0 = id3;
id5 = id0;
id6 = id1;
id7 = id8;
id9 = 5;
```

2. Συντακτική Ανάλυση

Ο συντακτικός αναλυτής με την σειρά του παίρνει σαν είσοδο τον κώδικα που του δίνει ο λεκτικός αναλυτής και ξεκινάει να θέτει μια ιεραρχική δομή στην ακολουθία ομάδα. Επομένως, αρχικά θα αναλύσει τις πρώτες 3 γραμμές που αφορούν την δήλωση μεταβλητών με τον παρακάτω τρόπο για κάθε μεταβλητή και την "**id2**" που αντιστοιχεί στο "**K2**" του source code:

Το σημείο κλειδί όμως είναι η γραμμή 4, όπου εκεί θα διαβάσει μετά το “**int**” το “**id3**”, που αντιστοιχεί στο “**K3**” του source code. Το “**K3**” όμως, με βάση τον ορισμό του αλφαβήτου μας για τα ονόματα των μεταβλητών, δεν είναι αποδεκτό όνομα για μεταβλητή, καθώς το “**3**” δεν ανήκει στο αλφάβητο μας. Αυτο σημαίνει ότι η συντακτική ανάλυση, δεν θα παράξει ποτέ κωδικά για την επόμενη φάση της μεταγλώττισης, που είναι η Σημασιολογική Ανάλυση και θα τερματίσει το πρόγραμμα με error προς τον χρήστη και αντίστοιχο μήνυμα για το σφάλμα του κατα τον ορισμό της μεταβλητής “**K3**”.

3. Σημασιολογική Ανάλυση

Η σημασιολογική ανάλυση δεν θα ξεκινήσει ποτέ καθώς το πρόγραμμά μας έχει σταματήσει στην λεκτική ανάλυση και έχει επιστραφεί error.