

Θεμα 3

A) i)

Όνομα Φοιτητή 1: **FOTIS TSIUMAS**

A.M. Φοιτητή 1: ΜΠΠΛ**21079**

Όνομα Φοιτητή 2: **KONSTANTINOS PETROU**

A.M. Φοιτητή 2: ΜΠΠΛ**21062**

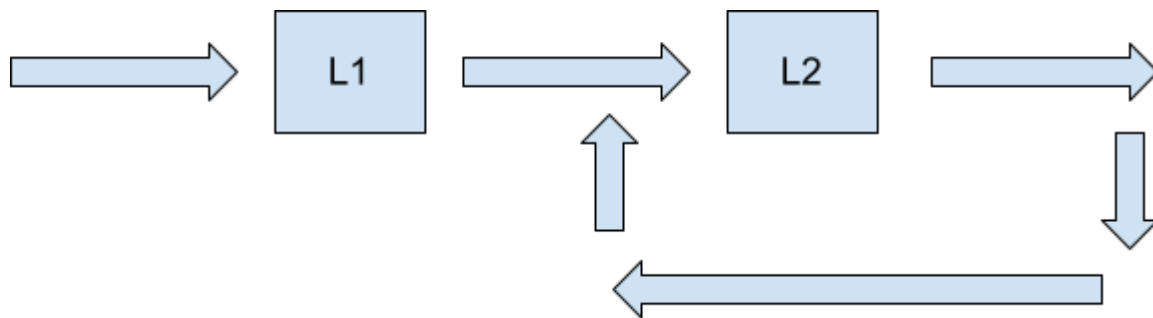
Alphabet: $\Sigma = \{A, E, F, I, K, M, O, P, R, S, T, U, 0, 1, 2, 6, 7, 9\}$

A) ii)

Regular Expression: **$L1L2^*$**

όπου **$L1$** = {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, salary, fullname κτλ) κ.α. Επίσης, υπάρχουν τα lexical values που έχουν εγχωρηθεί στους identifiers. 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:

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"

Να σημειωθεί εδώ ότι το A, A1 δεν τα αναγνωρίζει ως ονόματα μεταβλητών αλλά ως string, καθώς δεν δηλώνονται στην αρχή μαζί με τον τύπο τους. Αφού εισαχθούν

αυτά τα lexemes και αναλυθούν, δημιουργείται ένας πίνακας συμβόλων από τις παρακάτω ομάδες:

Ομάδα 1: int	Ομάδα 2: if	Ομάδα 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 20: id8	Ομάδα 20: id9		

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 A > A1 {  
    printf("K is greater than K1");  
}  
id1 = id4;  
id0 = id3;  
id5 = id0;  
id6 = id1;  
id7 = id8;  
id9 = 5;
```

2. Συνακτική 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 10 γραμμές που αφορούν την δήλωση μεταβλητών με τον παρακάτω τρόπο (ομοίως id0-id9):



Το σημείο κλειδί όμως είναι η γραφή 11 όπου εκεί θα διαβάσει το “If” και μετά θα περιμένει άνοιγμα παρένθεσης ώστε να ξεκινήσει η συνθήκη. Το πρώτο πράγμα όμως που θα βρει είναι την μεταβλητή id0 αρα θα σταματήσει εκεί το compilation και θα επιστρέψει σφάλμα τύπου **“error: expected ‘(’ before ‘id0’”**. Αυτό σημαίνει ότι δεν θα παράξει ποτέ κωδικά για την επόμενη φάση της μεταγλώττισης που είναι η Σημασιολογική Ανάλυση.

3. Σημασιολογική Ανάλυση

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