Πανεπιστήμιο Δυτικής Αττικής
Σχολή Μηχανικών
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Μεταγλωττιστές

1ο Μέρος Εργασίας (Α3)

Τμήμα Α3
Ομάδα (4)

ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ (ΠΑΔΑ-21390068-6ο εξάμηνο)
ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ (ΠΑΔΑ-20390191-8ο εξάμηνο)
ΜΠΗΛΙΩΝΗ ΠΑΡΑΣΚΕΥΗ (ΠΑΔΑ-20390286-8ο εξάμηνο)
ΤΗΛΕΜΑΧΟΣ ΠΟΥΛΙΑΝΑΣ (ΠΑΔΑ-21390304-6ο εξάμηνο)
ΑΘΑΝΑΣΙΟΣ ΜΟΥΤΖΟΥΡΗΣ (ΠΑΔΑ-21390137-6ο εξάμηνο)

ΑΘΗΝΑ

Περιεχόμενα

ΜΕΡΟΣ Α-3: Συμπλήρωση πρότυπου κώδικα FLEX

Παρουσίαση στόχου εργασίας

Ένας λεκτικός αναλυτής χρησιμοποιείται για την αναγνώριση των λεκτικών μονάδων. Οι λεκτικές μονάδες μπορεί να είναι διαχωριστικά λέξεων, χαρακτήρες, δεσμευμένες λέξεις, ακέραιοι, αριθμοί κινούμενης υποδιαστολής, σύμβολα και σχόλια. Κατανοώντας το πρόγραμμα simple-flex-code.l αντιλαμβάνεται κανείς με περισσότερη ευκολία την λειτουργία ενός τέτοιου αναλυτή και μπορεί να δημιουργήσει τον δικό του, με δικές του εντολές, σχόλια και να εμφανίζει αποτελέσματα με βάση τους κανόνες που θα οριστούν.

Σκοπός της παρούσας εργασίας είναι αρχικά η συμπλήρωση του προγράμματος simple-flex-code.l, η εκτέλεση και η κατανόηση του. Έπειτα, γίνεται η δημιουργία ενός λεκτικού αναλυτή ο οποίος αναγνωρίζει όλες τις λεκτικές μονάδες τα διαχωριστικά και τα σχόλια της γλώσσας UNI-C. Πιο συγκεκριμένα, αγνοεί τα white_spaces, αναγνωρίζει και αγνοεί τα σχόλια, όταν συναντάει χαρακτήρα αλλαγής γραμμής αυξάνει έναν μετρητή γραμμών για να εμφανίζει τη γραμμή της συμβολοσειράς που αναλύεται. Τέλος, όταν αναγνωρίζει μία λεκτική μονάδα επιτυχώς εμφανίζει την γραμμή του κώδικα, το αναγνωριστικό της όνομα καθώς και την τιμή της ενώ στην περίπτωση που η σάρωση εντοπίσει άγνωστη συμβολοσειρά εμφανίζει το μήνυμα " Line= αριθμός γραμμής, UNKNOWN TOKEN, value=”τιμή λανθασμένης συμβολοσειράς” " και συνεχίζει με την ανάλυση της επόμενης λέξης.

Αντιπαραβολή εισόδου – εξόδου & Αναλυτικός σχολιασμός

• Input: `Name`
Output:
```
Line = 1, token = IDENTIFIER, value = "Name"
```

Σχολιασμός:

Στο παράδειγμα (Name) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο με αποτελείται από κεφαλαίους και μικρούς λατινικούς χαρακτήρες (a-z, A-Z), άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “IDENTIFIER”. Ο μετρητής γραμμής έχει την τιμή “1” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `Name123`
Output:
```
Line = 2, token = IDENTIFIER, value = "Name123"
```

Σχολιασμός:

Στο παράδειγμα (Name123) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από κεφαλαίους και μικρούς λατινικούς χαρακτήρες (a-z, A-Z) και ψηφία 0 έως 9 (χωρίς να αρχίζει από αυτά), άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “IDENTIFIER”. Ο μετρητής γραμμής έχει την τιμή “2” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `Name_123`
Output:
```
Line = 3, token = IDENTIFIER, value = "Name_123"
```

Σχολιασμός:

Στο παράδειγμα (Name_123) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από κεφαλαίους και μικρούς λατινικούς χαρακτήρες (a-z, A-Z), το underscore(_) και ψηφία 0 έως 9 (όχι όμως στον αρχικό χαρακτήρα), άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “IDENTIFIER”. Ο μετρητής γραμμής έχει την τιμή “3” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `_name123`
Output:
```
Line = 4, token = IDENTIFIER, value = "_name123"
```

Σχολιασμός:

Στο παράδειγμα (_name123) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από το underscore(_), κεφαλαίους και μικρούς λατινικούς χαρακτήρες (a-z, A-Z) και ψηφία 0 έως 9 (όχι όμως στον αρχικό χαρακτήρα), άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “IDENTIFIER”. Ο μετρητής γραμμής έχει την τιμή “4” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `Name_1_END`
Output:
```
Line = 5, token = IDENTIFIER, value = "Name_1_END"
```

Σχολιασμός:

Στο παράδειγμα (Name_1_END) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από κεφαλαίους και μικρούς λατινικούς χαρακτήρες (a-z, A-Z), το underscore(_) και ψηφία 0 έως 9 (όχι όμως στον αρχικό χαρακτήρα), άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “IDENTIFIER”. Ο μετρητής γραμμής έχει την τιμή “5” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `name@email`
Output:
```
Line = 6, token = UNKNOWN_TOKEN, value = "name@email"
```

Σχολιασμός:

Στο παράδειγμα (name@email) του αρχείου input.txt, ο λεκτικός αναλυτής αρχίζει να αναγνωρίζει ένα αναγνωριστικό στο οποίο περιέχεται ο χαρακτήρας “@”, άρα δεν πληροί τις προϋποθέσεις και επιστρέφει “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “6” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `name_123!23`
Output:
```
Line = 7, token = UNKNOWN_TOKEN, value = "name_123!23"
```

Σχολιασμός:

Στο παράδειγμα (name_123!23) του αρχείου input.txt, ο λεκτικός αναλυτής αρχίζει να αναγνωρίζει ένα αναγνωριστικό στο οποίο περιέχεται ο χαρακτήρας “!”, άρα δεν πληροί τις προϋποθέσεις και επιστρέφει “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “7” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `name.123`
Output:
```
Line = 8, token = UNKNOWN_TOKEN, value = "name.123"
```

Σχολιασμός:

Στο παράδειγμα (name.123) του αρχείου input.txt, ο λεκτικός αναλυτής αρχίζει να αναγνωρίζει ένα αναγνωριστικό στο οποίο περιέχεται ο χαρακτήρας “.”, άρα δεν πληροί τις προϋποθέσεις και επιστρέφει “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “8” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `name\123`
Output:
```
Line = 9, token = UNKNOWN_TOKEN, value = "name\123"
```

Σχολιασμός:

Στο παράδειγμα (name\123) του αρχείου input.txt, ο λεκτικός αναλυτής αρχίζει να αναγνωρίζει ένα αναγνωριστικό στο οποίο περιέχεται ο χαρακτήρας “\”, άρα δεν πληροί τις προϋποθέσεις και επιστρέφει “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “9” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `123_name`
Output:
```
Line = 10, token = UNKNOWN_TOKEN, value = "123_name"
```

Σχολιασμός:

Στο παράδειγμα (123_name) του αρχείου input.txt, ο λεκτικός αναλυτής αρχίζει να αναγνωρίζει έναν αριθμό τον οποίο ακολουθεί το underscore (_), άρα δεν πληροί τις προϋποθέσεις για κάποια από τις επιλογές και επιστρέφει “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “10” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `"Test"`
Output:
```
Line = 11, token = STRING, value = ""Test""
```

Σχολιασμός:

Στο παράδειγμα (“Test”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό, το οποίο περικλείεται από διπλές αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “11” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `"Example using \"Quotes!\""`
Output:
```
Line = 12, token = STRING, value = ""Example using \"Quotes!\""
```

Σχολιασμός:

Στο παράδειγμα (“Example using \"Quotes!\"”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό, το οποίο περικλείεται από διπλές αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο. Η χρήση του backslash μαζί με τη διπλή απόστροφο \” είναι αποδεκτή εφόσον αναγνωρίζεται ως σειρά διαφυγής, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “12” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `"Newline \n"`
Output:
```
Line = 13, token = STRING, value = ""Newline \n""
```

Σχολιασμός:

Στο παράδειγμα (“Newline \n”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό, το οποίο περικλείεται από διπλές αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο. Η χρήση του \n παρόλο που συμβολίζει την αλλαγή γραμμής (newline) είναι αποδεκτή διότι αναγνωρίζεται ως σειρά διαφυγής και δεν εκτελείται ως αλλαγή γραμμής, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “13” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: " "
Output:
```
Line = 14, token = STRING, value = "" ""
```

Σχολιασμός:

Στο παράδειγμα (“ ”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό, το οποίο περικλείεται από διπλές αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “14” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: "example with tab and different characters %$#^&*"
Output:
```
Line = 15, token = STRING, value = ""example with tab and different
characters %$#^&*""
```

Σχολιασμός:

Στο παράδειγμα (“example with tab and different characters %$#^&*”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό, το οποίο περικλείεται από διπλές αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “15” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: "This is an example with \"
Output:
```
Line = 16, token = UNKNOWN_TOKEN, value = ""This"
Line = 16, token = IDENTIFIER, value = "is"
Line = 16, token = IDENTIFIER, value = "an"
Line = 16, token = IDENTIFIER, value = "example"
Line = 16, token = IDENTIFIER, value = "with"
Line = 16, token = UNKNOWN_TOKEN, value = "\""
```

Σχολιασμός:

Στο παράδειγμα (“This is an example with \”) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα λεκτικό κυριολεκτικό, όμως παρόλο που υπάρχουν διπλοί απόστροφοι ” στην αρχή της φράσης, δεν υπάρχουν στο τέλος, η χρήση του backslash μαζί με τη διπλή απόστροφο \” αναγνωρίζεται ως σειρά διαφυγής (escape sequence) καθιστώντας τη διπλή απόστροφο μη λειτουργική. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως UNKNOWN_TOKEN το ““This”, ως “IDENTIFIER” τα “is”, “an”, “example”, “with” επειδή πληρούν τις προϋποθέσεις ενός αναγνωριστικού και ως “UNKNOWN_TOKEN” το “\””. Ο μετρητής γραμμής έχει την τιμή “16” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: `"Trying two strings in one line""This is another string"`
Output:

```
Line = 17, token = STRING, value = ""Trying two strings in one line""
Line = 17, token = STRING, value = ""This is another string""
```

Σχολιασμός:

Στο παράδειγμα (“Trying two strings in one line""This is another string”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει δύο λεκτικά κυριολεκτικά τα οποία περικλείονται από διπλές “ αποστρόφους και δεν περιλαμβάνουν backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο, άρα πληρούν τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “17” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: `"Using space character after closing the string"`
Output:

```
Line = 18, token = STRING, value = ""Using space character after closing the string""
```

Σχολιασμός:

Στο παράδειγμα (“Using space character after closing the string") του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα λεκτικό κυριολεκτικό ή συμβολοσειρά (STRING), το οποίο περικλείεται από διπλές “ αποστρόφους και δεν περιλαμβάνει backslash (\), νέα γραμμή (\n) ή διπλή απόστροφο, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “STRING”. Ο μετρητής γραμμής έχει την τιμή “18” και το μήνυμα που εμφανίζεται είναι σωστό, παρόλο που μετά την διπλή απόστροφο “ στο τέλος του string υπάρχει το κενό (space).

• Input: `"something" something "something"`
Output:

```
Line = 19, token = STRING, value = ""something""
Line = 19, token = IDENTIFIER, value = "something"
Line = 19, token = STRING, value = ""something""
```

Σχολιασμός:

Στο παράδειγμα (“something" something "something”) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ως λεκτικά κυριολεκτικά την πρώτη και την τελευταία λέξη, οι οποίες περικλείονται από διπλά εισαγωγικά “, άρα τα επιστρέφει ως “STRING”. Η ενδιάμεση λέξη επιστρέφεται από τον λεκτικό αναλυτή ως “IDENTIFIER” αφού πληροί τις προϋποθέσεις ενός αναγνωριστικού. Ο μετρητής γραμμής έχει την τιμή “19” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: "Wrong input
Output:
```
Line = 20, token = UNKNOWN_TOKEN, value = ""Wrong"
Line = 20, token = IDENTIFIER, value = "input"
```

Σχολιασμός:

Στο παράδειγμα (““Wrong input”) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα λεκτικό κυριολεκτικό, όμως παρόλο που υπάρχουν διπλοί απόστροφοι ” στην αρχή της φράσης, δεν υπάρχουν στο τέλος. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το ““Wrong” και ως “IDENTIFIER” το “input” επειδή πληροί τις προϋποθέσεις ενός αναγνωριστικού. Ο μετρητής γραμμής έχει την τιμή “20” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: 0
Output:
```
Line = 21, token = INTEGER, value = "0"
```

Σχολιασμός:

Στο παράδειγμα (0) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν οκταδικό ακέραιο, το 0, άρα πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “INTEGER”. Ο μετρητής γραμμής έχει την τιμή “21” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: 214748
Output:
```
Line = 22, token = INTEGER, value = "214748"
```

Σχολιασμός :

Στο παράδειγμα (214748) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα δεκαδικό ακέραιο αριθμό ο οποίος ξεκινάει με ένα μη μηδενικό ψηφίο (1-9) και ακολουθείτε από δεκαδικά ψηφία (0-9). Επομένως πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “INTEGER“. Ο μετρητής γραμμής έχει την τιμή “22” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: 0x4F
Output:
```
Line = 23, token = INTEGER, value = "0x4F"
```

Σχολιασμός:

Στο παράδειγμα (0x4F) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα δεκαεξαδικό ακέραιο αριθμό ο οποίος ξεκινάει με 0x και ακολουθείτε από δύο δεκαεξαδικά ψηφία (4, F). Επομένως πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “INTEGER“. Ο μετρητής γραμμής έχει την τιμή “23” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0XFF42`
Output:
```
Line = 24, token = INTEGER, value = "0XFF42"
```

Σχολιασμός:

Στο παράδειγμα (0XFF42) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα δεκαεξαδικό ακέραιο αριθμό ο οποίος ξεκινάει με 0X και ακολουθείτε από τέσσερα δεκαεξαδικά ψηφία (FF42). Επομένως πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “INTEGER“. Ο μετρητής γραμμής έχει την τιμή “24” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0147`
Output:
```
Line = 25, token = INTEGER, value = "0147"
```

Σχολιασμός:

Στο παράδειγμα (0147) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν οκταδικό ακέραιο αριθμό ο οποίος ξεκινάει με τον αριθμό 0 και ακολουθείτε από τρία οκταδικά ψηφία (147). Επομένως πληροί τις προϋποθέσεις και ο λεκτικός αναλυτής επιστρέφει “INTEGER“. Ο μετρητής γραμμής έχει την τιμή “25” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `029123`
Output:
```
Line = 26, token = UNKNOWN_TOKEN, value = "029123"
```

Σχολιασμός:

Στο παράδειγμα (029123) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα οκταδικό ακέραιο αριθμό, αλλά παρόλο που ξεκινάει με τον αριθμό 0, ακολουθείτε από το (9), το οποίο ξεπερνάει τα όρια ένος οκταδικού αριθμού (0-7). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “029123”. Ο μετρητής γραμμής έχει την τιμή “26” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0xG123`
Output:
```
Line = 27, token = UNKNOWN_TOKEN, value = "0xG123"
```

Σχολιασμός:

Στο παράδειγμα (0xG123) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα δεκαεξαδικό ακέραιο αριθμό, αλλά παρόλο που ξεκινάει με 0x, ακολουθείτε από το (G), το οποίο ξεπερνάει τα όρια ένος δεκαεξαδικού αριθμού (0-9, A-F). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “0xG123”. Ο μετρητής γραμμής έχει την τιμή “27” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0XX15`
Output:
```
Line = 28, token = UNKNOWN_TOKEN, value = "0XX15"
```

Σχολιασμός:

Στο παράδειγμα (0XX15) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα δεκαεξαδικό ακέραιο αριθμό, αλλά παρόλο που ξεκινάει με 0X, ακολουθείτε από το (X), το οποίο ξεπερνάει τα όρια ένος δεκαεξαδικού αριθμού (0-9, A-F). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “0XX15”. Ο μετρητής γραμμής έχει την τιμή “28” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `055X3`
Output:
```
Line = 29, token = UNKNOWN_TOKEN, value = "055X3"
```

Σχολιασμός:

Στο παράδειγμα (055X3) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει έναν οκταδικό ακέραιο, αλλά παρόλο που ξεκινάει με τον αριθμό 0, ακολουθείτε από το (X), το οποίο ξεπερνάει τα όρια ένος οκταδικού αριθμού (0-7). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “055X3”. Ο μετρητής γραμμής έχει την τιμή “29” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0x`
Output:
```
Line = 30, token = UNKNOWN_TOKEN, value = "0x"
```

Σχολιασμός:

Στο παράδειγμα (0x) του αρχείου input.txt, ο λεκτικός αναλυτής θα έπρεπε να αναγνωρίζει ένα δεκαεξαδικό ακέραιο αριθμό, αλλά παρόλο που ξεκινάει με 0x, δεν ακολουθείτε έπειτα από ένα ή περισσότερα δεκαεξαδικά ψηφία (0-9, A-F). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “0x”. Ο μετρητής γραμμής έχει την τιμή “30” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `3.14`
Output:
```
Line = 31, token = FLOAT, value = "3.14"
```

Σχολιασμός:

Στο παράδειγμα (3.14) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν αριθμό κινούμενης υποδιαστολής, που το ακέραιο (3) και το δεκαδικό (14) μέρος του διαχωρίζονται με τελεία “.”.). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “FLOAT” το “3.14”. Ο μετρητής γραμμής έχει την τιμή “31” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `14e1.2`
Output:
```
Line = 32, token = FLOAT, value = "14e1.2"
```

Σχολιασμός:

Στο παράδειγμα (14e1.2) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν αριθμό κινούμενης υποδιαστολής. Ο αριθμός αυτός αποτελείται από το ακέραιο μέρος (14), την ύψωση σε δύναμη (e) και τον αριθμό για την δύναμη (1.2). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “FLOAT” το “14e1.2”. Ο μετρητής γραμμής έχει την τιμή “32” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0e-1.2`
Output:
```
Line = 33, token = FLOAT, value = "0e-1.2"
```

Σχολιασμός:

Στο παράδειγμα (0e-1.2) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν αριθμό κινούμενης υποδιαστολής Ο αριθμός αυτός αποτελείται από το ακέραιο μέρος (0), την ύψωση σε δύναμη (e) και τον αριθμό για την δύναμη (-1.2). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “FLOAT” το “0e-1.2”. Ο μετρητής γραμμής έχει την τιμή “33” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `0e0`
Output:
```
Line = 34, token = FLOAT, value = "0e0"
```

Σχολιασμός:

Στο παράδειγμα (0e0) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν αριθμό κινούμενης υποδιαστολής. Ο αριθμός αυτός αποτελείται από το ακέραιο μέρος (0), την ύψωση σε δύναμη (e) και τον αριθμό για την δύναμη (0). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “FLOAT” το “0e0”. Ο μετρητής γραμμής έχει την τιμή “34” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `3.1e1`
Output:
```
Line = 35, token = FLOAT, value = "3.1e1"
```

Σχολιασμός:

Στο παράδειγμα (3.1e1) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει έναν αριθμό κινούμενης υποδιαστολής. Ο αριθμός αυτός αποτελείται από το ακέραιο μέρος (3), την τελεία (.), το δεκαδικό μέρος (1), την ύψωση σε δύναμη (e) και τον αριθμό για την δύναμη (1). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “FLOAT” το “3.1e1”. Ο μετρητής γραμμής έχει την τιμή “35” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `.10`
Output:
```
Line = 36, token = UNKNOWN_TOKEN, value = ".10"
```

Σχολιασμός:

Στο παράδειγμα (.10) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα χαρακτήρα άγνωστης συμβολοσειράς καθώς ξεκινάει με τη τελεία “.” το οποίο δεν είναι αντιστοιχεί σε καμία λεκτική μονάδα. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως

“UNKNOWN_TOKEN” το “.10”. Ο μετρητής γραμμής έχει την τιμή “36” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: e22
Output:
```
Line = 37, token = IDENTIFIER, value = "e22"
```

Σχολιασμός:

Στο παράδειγμα (e22) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από ένα μικρό λατινικό χαρακτήρα (e) και 2 ψηφία (2) και (2). Στη προκειμένη περίπτωση το “e” δεν αναγνωρίζεται ως ύψωση σε δύναμη καθώς πριν από αυτό δεν προηγείται κάποιος ακέραιος ή δεκαδικός αριθμός. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “IDENTIFIER” το “e22”. Ο μετρητής γραμμής έχει την τιμή “37” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: 3-
Output:
```
Line = 38, token = UNKNOWN_TOKEN, value = "3-"
```

Σχολιασμός:

Στο παράδειγμα (3-) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα χαρακτήρα άγνωστης συμβολοσειράς, καθώς αποτελείται από έναν ακέραιο αριθμό (3) που ακολουθείτε από το “-” που δεν είναι αποδεκτό χωρίς να υπάρχει κενό ανάμεσα στους χαρακτήρες (space). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “3-”. Ο μετρητής γραμμής έχει την τιμή “38” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: 1.10.10
Output:
```
Line = 39, token = UNKNOWN_TOKEN, value = "1.10.10"
```

Σχολιασμός:
Στο παράδειγμα (1.10.10) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα χαρακτήρα άγνωστης συμβολοσειράς, καθώς περιλαμβάνει περισσότερες από μία τελείες “.” το οποίο δεν είναι αποδεκτό σύμφωνα με τους κανόνες των δεκαδικών αριθμών. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “1.10.10”. Ο μετρητής γραμμής έχει την τιμή “39” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: 0e
Output:
```
Line = 40, token = UNKNOWN_TOKEN, value = "0e"
```

Σχολιασμός:

Στο παράδειγμα (0e) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα χαρακτήρα άγνωστης συμβολοσειράς, καθώς το “e”, σύμφωνα με τους κανόνες των αριθμών πρέπει να προηγείται και να ακολουθείτε από έναν τουλάχιστον ακέραιο αριθμό. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “UNKNOWN_TOKEN” το “0e”. Ο μετρητής γραμμής έχει την τιμή “40” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `continue`
Output:
```
Line = 41, token = KEYWORD, value = "continue"
```

Σχολιασμός:

Στο παράδειγμα (continue) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει μία λέξη κλειδί, η οποία είναι καταχωρημένη στον πίνακα συμβόλων και συνεπώς θεωρείται δεσμευμένη. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως "KEYWORD" το `"continue".` Ο μετρητής γραμμής έχει την τιμή "41" και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `long`
Output:
```
Line = 42, token = KEYWORD, value = "long"
```

Σχολιασμός:

Στο παράδειγμα (long) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει μία λέξη, η οποία είναι καταχωρημένη στον πίνακα συμβόλων και συνεπώς θεωρείται δεσμευμένη. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως "KEYWORD" το `"long".` Ο μετρητής γραμμής έχει την τιμή "42" και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `break`
Output:
```
Line = 43, token = KEYWORD, value = "break"
```

Σχολιασμός:

Στο παράδειγμα (break) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει μία λέξη, η οποία είναι καταχωρημένη στον πίνακα συμβόλων και συνεπώς θεωρείται δεσμευμένη. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως "KEYWORD" το `"break".` Ο μετρητής γραμμής έχει την τιμή "43" και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `struct`
Output:
```
Line = 44, token = KEYWORD, value = "struct"
```

Σχολιασμός :

Στο παράδειγμα (struct) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει μία λέξη, η οποία είναι καταχωρημένη στον πίνακα συμβόλων και συνεπώς θεωρείται δεσμευμένη. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως "KEYWORD" το `"struct".` Ο μετρητής γραμμής έχει την τιμή "44" και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `switch`
Output:
```
Line = 45, token = KEYWORD, value = "switch"
```

Σχολιασμός:

Στο παράδειγμα (switch) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει μία λέξη, η οποία είναι καταχωρημένη στον πίνακα συμβόλων και συνεπώς θεωρείται δεσμευμένη. Επομένως, ο λεκτικός αναλυτής επιστρέφει ως "KEYWORD" το `"switch".` Ο μετρητής γραμμής έχει την τιμή "45" και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `1 + 2 = 3`
Output:

```
Line = 46, token = INTEGER, value = "1"
Line = 46, token = OPERATOR, value = "+"
Line = 46, token = INTEGER, value = "2"
Line = 46, token = OPERATOR, value = "="
Line = 46, token = INTEGER, value = "3"
```

Σχολιασμός:

Στο παράδειγμα (1 + 2 = 3) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τρεις δεκαδικούς ακέραιους αριθμούς (1, 2, 3). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και δύο τελεστές(+, =), οι οποίοι είναι δεσμευμένοι χαρακτήρες στη γλώσσα UNI-C. Ο μετρητής γραμμής έχει την τιμή “46” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (INTEGER) – (OPERATOR) που είναι και σωστό.

• Input: `10 - 4 = 6`
Output:

```
Line = 47, token = INTEGER, value = "10"
Line = 47, token = OPERATOR, value = "-"
Line = 47, token = INTEGER, value = "4"
Line = 47, token = OPERATOR, value = "="
Line = 47, token = INTEGER, value = "6"
```

Σχολιασμός:

Στο παράδειγμα (10 - 4 = 6) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τρεις δεκαδικούς ακέραιους αριθμούς (10, 4, 6). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και δύο τελεστές(-, =), οι οποίοι είναι δεσμευμένοι χαρακτήρες στη γλώσσα UNI-C. Ο μετρητής γραμμής έχει την τιμή “47” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (INTEGER) – (OPERATOR) που είναι και σωστό.

• Input: `13 * 4 =5 2`
Output:

```
Line = 48, token = INTEGER, value = "13"
Line = 48, token = OPERATOR, value = "*"
Line = 48, token = INTEGER, value = "4"
Line = 48, token = UNKNOWN_TOKEN, value = "=5"
Line = 48, token = INTEGER, value = "2"
```

Σχολιασμός:

Στο παράδειγμα (13 * 4 = 5 2) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τρεις δεκαδικούς ακέραιους αριθμούς (13, 4, 2). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και ένα τελεστή(*), ο οποίος είναι δεσμευμένος χαρακτήρας στη γλώσσα UNI-C. Τέλος αναγνωρίζει και έναν χαρακτήρα άγνωστης συμβολοσειράς (=5) το οποίο δεν αντιστοιχεί σε καμία λεκτική μονάδα, οπότε χαρακτηρίζεται ως “UNKNOWN_TOKEN”. Ο μετρητής γραμμής έχει την τιμή “48” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (INTEGER) – (OPERATOR) που είναι και σωστό.

• Input: `10 ! = 3628800`
Output:
```
Line = 49, token = INTEGER, value = "10"
Line = 49, token = OPERATOR, value = "!"
Line = 49, token = OPERATOR, value = "="
Line = 49, token = INTEGER, value = "3628800"
```

Σχολιασμός:

Στο παράδειγμα (10 ! =3628800) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει δύο δεκαδικούς ακέραιους αριθμούς (10, 3628800). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και δύο τελεστές(!, =), οι οποίοι είναι δεσμευμένοι χαρακτήρες στη γλώσσα UNI-C. Ο μετρητής γραμμής έχει την τιμή “49” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (INTEGER) – (OPERATOR) που είναι και σωστό.

• Input: `55 / 5 = 11`
Output:
```
Line = 50, token = INTEGER, value = "55"
Line = 50, token = OPERATOR, value = "/"
Line = 50, token = INTEGER, value = "5"
Line = 50, token = OPERATOR, value = "="
Line = 50, token = INTEGER, value = "11"
```

Σχολιασμός:

Στο παράδειγμα (55 / 5 = 11) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τρεις δεκαδικούς ακέραιους αριθμούς (55, 5, 11). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και δύο τελεστές(/, =), οι οποίοι είναι δεσμευμένοι χαρακτήρες στη γλώσσα UNI-C. Ο μετρητής γραμμής έχει την τιμή “50” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (INTEGER) – (OPERATOR) που είναι και σωστό.

• Input: `// comments`
Output:
```
(κενό)
```

Σχολιασμός:

Στο παράδειγμα (// comments) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους διπλούς χαρακτήρες “/” ως σχόλια και ολοκληρώνονται με την αλλαγή γραμμής (\n). Ο μετρητής γραμμής δεν αυξάνεται καθώς τα σχόλια αγνοούνται και στην οθόνη δεν εμφανίζεται κάποιο μήνυμα ως output.

• Input: `/*comments*/`
Output:
```
(κενό)
```

Σχολιασμός:

Στο παράδειγμα (/*comments*/) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους χαρακτήρες “/*” και “*/” στην αρχή και στο τέλος της συμβολοσειράς αντίστοιχα ως σχόλια. Ο μετρητής γραμμής δεν αυξάνεται καθώς τα σχόλια αγνοούνται και στην οθόνη δεν εμφανίζεται κάποιο μήνυμα ως output.

• Input: `/**/`
Output:
`(κενό)`

Σχολιασμός:

Στο παράδειγμα (/**/) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους χαρακτήρες ‘/*’ και ‘*/’ στην αρχή και στο τέλος της συμβολοσειράς αντίστοιχα ως σχόλια. Ο μετρητής γραμμής δεν αυξάνεται καθώς τα σχόλια αγνοούνται και στην οθόνη δεν εμφανίζεται κάποιο μήνυμα ως output.

• Input: `/* comments */ test`
Output:
`Line = 51, token = IDENTIFIER, value = "test"`

Σχολιασμός:

Στο παράδειγμα (/* comments */ test) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους χαρακτήρες ‘/*’ και ‘*/’ στην αρχή και στο τέλος της συμβολοσειράς αντίστοιχα ως σχόλια. Έπειτα αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από τέσσερις μικρούς λατινικούς χαρακτήρες (test). Επομένως, ο λεκτικός αναλυτής επιστρέφει ως “`IDENTIFIER`” το “`test`“ και το σχόλιο αγνοείται. Ο μετρητής γραμμής έχει την τιμή “51” και το μήνυμα που εμφανίζεται είναι σωστό.

• Input: `//`
Output:
`(κενό)`

Σχολιασμός:

Στο παράδειγμα (//) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους διπλούς χαρακτήρες ‘/’ ως σχόλια και ολοκληρώνονται με την αλλαγή γραμμής (\n). Ο μετρητής γραμμής δεν αυξάνεται καθώς τα σχόλια αγνοούνται και στην οθόνη δεν εμφανίζεται κάποιο μήνυμα ως output.

• Input (56-58): `/*comment`
`with enter`
`*/`
Output:
`(κενό)`

Σχολιασμός:

Στο παράδειγμα

(/*comment

With enter

/) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει τους χαρακτήρες ‘/’ και ‘*/’ στην αρχή και στο τέλος της συμβολοσειράς αντίστοιχα ως σχόλια ανεξάρτητα από τον αριθμό των γραμμών που μεσολαβούν. Ο μετρητής γραμμής δεν αυξάνεται καθώς τα σχόλια αγνοούνται και στην οθόνη δεν εμφανίζεται κάποιο μήνυμα ως output.

• Input (space):
Output:
```
(κενό)
```

Σχολιασμός:

Στο παράδειγμα ( ) (space) του αρχείου input.txt, ο λεκτικός αναλυτής αγνοεί το space. Ο μετρητής γραμμής αυξάνεται παίρνοντας την τιμή “52” αλλά στην οθόνη δεν εμφανίζεται κάποιο μήνυμα.

• Input (tab):
Output:
```
(κενό)
```

Σχολιασμός:

Στο παράδειγμα ( ) (tab) του αρχείου input.txt, ο λεκτικός αναλυτής αγνοεί το tab. Ο μετρητής γραμμής αυξάνεται παίρνοντας την τιμή “53” αλλά στην οθόνη δεν εμφανίζεται κάποιο μήνυμα.

• Input: `check !`
Output:
```
Line = 54, token = IDENTIFIER, value = "check"
Line = 54, token = OPERATOR, value = "!"
```

Σχολιασμός:

Στο παράδειγμα (check !) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει ένα αναγνωριστικό, το οποίο αποτελείται από πέντε μικρούς λατινικούς χαρακτήρες (check). Επίσης, ο λεκτικός αναλυτής αναγνωρίζει και ένα τελεστή(!), ο οποίος είναι δεσμευμένος χαρακτήρας στη γλώσσα UNI-C. Ο μετρητής γραμμής έχει την τιμή “54” για κάθε ένα στοιχείο που εντοπίζει και εμφανίζει το κατάλληλο μήνυμα (IDENTIFIER) – (OPERATOR).

• Input: `This is a sentence with spaces!`
Output:
```
Line = 55, token = IDENTIFIER, value = "This"
Line = 55, token = IDENTIFIER, value = "is"
Line = 55, token = IDENTIFIER, value = "a"
Line = 55, token = IDENTIFIER, value = "sentence"
Line = 55, token = IDENTIFIER, value = "with"
Line = 55, token = UNKNOWN_TOKEN, value = "spaces!"
```

Σχολιασμός:

Στο παράδειγμα (This is a sentence with spaces!) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει πέντε identifiers (This,is ,a, sentence, with), καθώς πληρούν τις προϋποθέσεις και επιστρέφονται από τον λεκτικό αναλυτή ως “IDENTIFIERS”. Αυτό συμβαίνει επειδή ο λεκτικός αναλυτής αγνοεί τους space χαρακτήρες. Ο λεκτικός αναλυτής δεν αναγνωρίζει το “spaces!” επειδή αρχικά το αναγνωρίζει ως αναγνωριστικό, όμως ακολουθεί ο χαρακτήρας “!” και έτσι δεν πληροί τις προϋποθέσεις, άρα επιστρέφεται ως “UKNOWN_TOKEN”.Ο μετρητής γραμμής έχει την τιμή “55” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: `Ok, now tab	!`
Output:

```
Line = 56, token = UNKNOWN_TOKEN, value = "Ok,"
Line = 56, token = IDENTIFIER, value = "now"
Line = 56, token = IDENTIFIER, value = "tab"
Line = 56, token = OPERATOR, value = "!"
```

Σχολιασμός:

Στο παράδειγμα (Ok, now tab !) του αρχείου input.txt, ο λεκτικός αναλυτής δεν αναγνωρίζει το (Ok,) επειδή υπάρχει ο χαρακτήρας “,” που δεν αναγνωρίζεται από κάποια από τις επιλογές του κώδικα, οπότε ο λεκτικός αναλυτής επιστρέφει “UKNOWN_TOKEN”. Τα “now” και “tab” επιστρέφονται ως “IDENTIFIER” καθώς πληρούν τις προϋποθέσεις των αναγνωριστικών. Ο χαρακτήρας “!” επιστρέφεται ως “OPERATOR” επειδή είναι δεσμευμένος χαρακτήρας της Uni-C. Ο λεκτικός αναλυτής αγνοεί τα whitespaces (χαρακτήρες space και tab). Ο μετρητής γραμμής έχει την τιμή “56” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

• Input: `Seperating ; words ; with ; space semi colon and	tab	;`
Output:

```
Line = 57, token = IDENTIFIER, value = "Seperating"
Line = 57, token = IDENTIFIER, value = "words"
Line = 57, token = IDENTIFIER, value = "with"
Line = 57, token = IDENTIFIER, value = "space"
Line = 57, token = IDENTIFIER, value = "semi"
Line = 57, token = IDENTIFIER, value = "colon"
Line = 57, token = IDENTIFIER, value = "and"
Line = 57, token = IDENTIFIER, value = "tab"
```

Σχολιασμός:

Στο παράδειγμα (Seperating ; words ; with ; space semi colon and tab ;) του αρχείου input.txt, ο λεκτικός αναλυτής αναγνωρίζει οκτώ αναγνωριστικά (Seperating, words, with, space, semi, colon, and, tab), καθώς πληρούν τις προϋποθέσεις και επιστρέφονται από τον λεκτικό αναλυτή ως “IDENTIFIERS”. Αυτό συμβαίνει επειδή ο λεκτικός αναλυτής αγνοεί τα whitespaces (χαρακτήρες space, tab) και το semi colon (;). Ο μετρητής γραμμής έχει την τιμή “57” για κάθε ένα στοιχείο που εντοπίζει και τα μηνύματα που εμφανίζει είναι σωστά.

Αναφορές / Δυσκολίες

Το simple-flex-code.l που μας δόθηκε ήταν αρκετά περιεκτικό και τεκμηριωμένο (αν και με greeklish). Συμπληρώσαμε στον κώδικα τις κανονικές εκφράσεις που είχαμε δημιουργήσει από το (α-2) μέρος της εργασίας και κάναμε τις απαραίτητες αλλαγές. Δημιουργήσαμε τις κανονικές εκφράσεις για keywords και operators που αν και δεν μας δυσκόλεψαν, μας φανήκαν σχετικά ”απλές” σε σχέση με τις υπόλοιπες. Στη συνέχεια, τις εκφράσεις για comments, whitespaces και unknown tokens. Επιπλέον, αλλάξαμε την δομή ελέγχου για τα arguments ώστε να καλύπτουν κάθε πιθανή περίπτωση και να εμφανίζουν τα κατάλληλα μηνύματα. Επίσης, το flex, δεν αντιλαμβάνεται το /d στις κανονικές εκφράσεις, οπότε έπρεπε να το αλλάξουμε όπου το χρησιμοποιήσαμε με το [0-9] για τους αριθμούς. Τέλος, προσθέσαμε στον αρχικό κώδικα τα fclose(), ώστε σε περίπτωση που ανοίγει κάποιο αρχείο με fopen(), να κλείνει.

Δεν αντιμετωπίσαμε ιδιαίτερες δυσκολίες. Αναφέρουμε όμως, πως αν και ο κώδικας λειτουργεί βάσει της εκφώνησης, στην γραμματική της γλώσσας Uni-C, η εκτύπωση ορισμένων λεκτικών μονάδων διαφέρει (π.χ. integer που θα πρέπει να είναι integer(value)). Σε περίπτωση που θα έπρεπε να είναι έτσι, θα προσθέταμε μια δομή ελέγχου if μέσα στην επαναληπτική δομή while ώστε ανάλογα με τον τύπο του token, να γίνεται η αντίστοιχη εκτύπωση του αποτελέσματος (π.χ. αν είναι ακέραιος θα πρέπει να εκτυπώνεται και η τιμή δίπλα στον τύπο «INTEGER(τιμή)» ενώ σε άλλους όχι (π.χ. αν είναι αριθμός κινούμενης υποδιαστολής «FLOAT»). Επιπλέον, αναφέρουμε ότι για την αναγνώριση των τελεστών, θα πρέπει να υπάρχουν κενά ανάμεσά σε αυτούς και των χαρακτήρων (π.χ. 2 + 5 όχι 2+5), όπως και για το διαχωριστικό (;) ισχύει το ίδιο.

Λειτουργία:

Το αρχείο 4-flex-code.l κάνει compile και λειτουργεί χωρίς σφάλματα. Μετά από εξαντλητική χρήση λάβαμε τα επιθυμητά αποτελέσματα. Το αρχείο εισόδου (input.txt) είναι σχεδιασμένο έτσι, ώστε να καλύπτει πλήρως τη λειτουργικότητα του κώδικα, αφού εμπεριέχονται παραδείγματα με όλες τις πιθανές εισόδους. Ως ομάδα, δεν βρήκαμε κάποια έλλειψη στην εργασία μας.

Προγράμματα για την εκπόνηση της εργασίας:

Χρησιμοποιήσαμε διάφορα προγράμματα για την συγκεκριμένη εργασία που αναφέρονται παρακάτω:

- Microsoft Word
- Flex
- Visual Studio

Ανάθεση αρμοδιοτήτων – ρόλων

Παρουσίαση στόχου εργασίας: ΜΟΥΤΖΟΥΡΗΣ ΑΘΑΝΑΣΙΟΣ

Κώδικας FLEX: ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ

Οριστικοποίηση κώδικα FLEX:
ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ, ΜΠΗΛΙΩΝΗ ΠΑΡΑΣΚΕΥΗ

Σχολιασμός (εντός) του κώδικα FLEX (αρχείο 4-flex-code.l):
ΜΠΗΛΙΩΝΗ ΠΑΡΑΣΚΕΥΗ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ

Σχολιασμός (εντός) του αρχείου header (token.h):
ΜΠΗΛΙΩΝΗ ΠΑΡΑΣΚΕΥΗ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ

Σχολιασμός αποτελεσμάτων:
ΜΟΥΤΖΟΥΡΗΣ ΑΘΑΝΑΣΙΟΣ, ΠΟΥΛΙΑΝΑΣ ΤΗΛΕΜΑΧΟΣ,
ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ

Αναφορές / Δυσκολίες: ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ

Λειτουργία: ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ

Επιμέλεια εγγράφου τεκμηρίωσης:
ΘΩΜΑΣ ΝΙΚΟΛΑΟΣ, ΠΛΑΓΟΥ ΑΙΚΑΤΕΡΙΝΗ