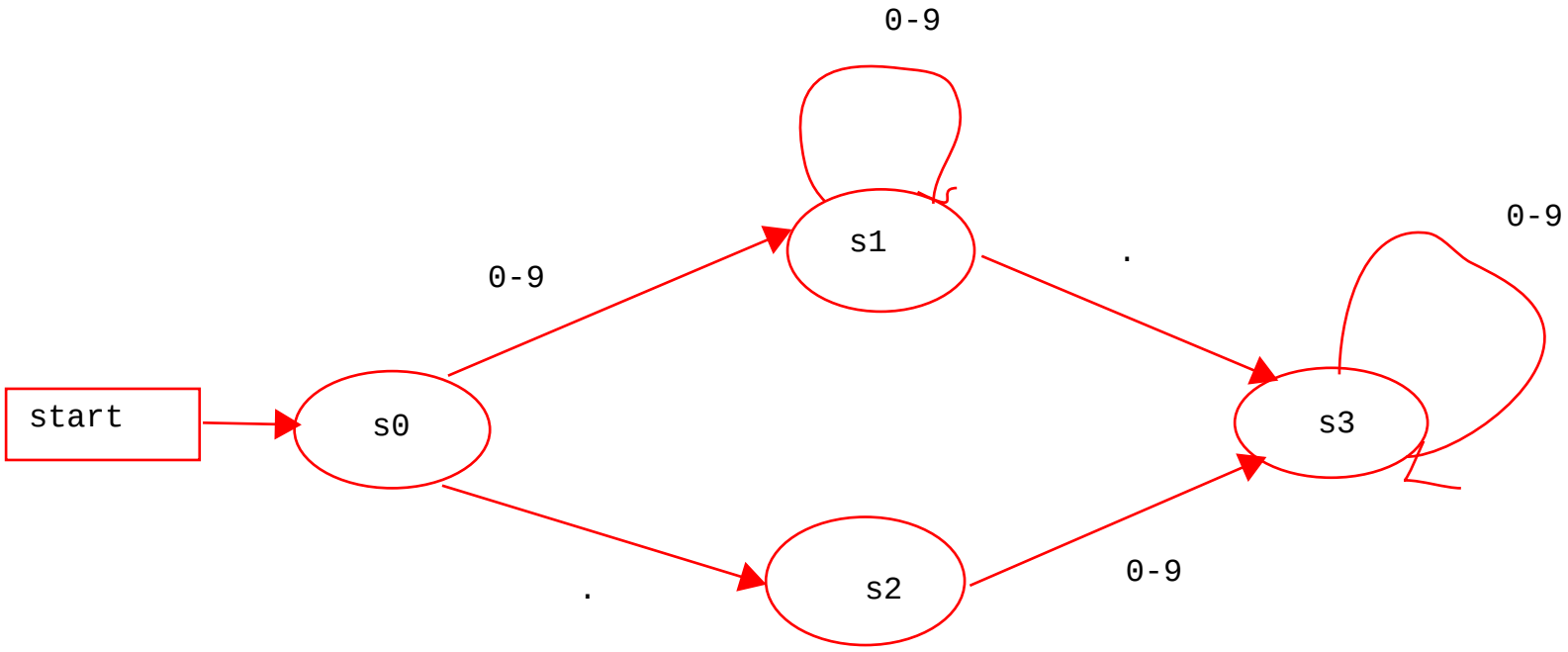
Μεταγλωτιστές 2020
Προγραμματιστική Εργασία #1

Ονοματεπώνυμο: ΚΡΙΣΤΙΑΝ ΛΕΚΑ
ΑΜ: Π2017153

Αναφορά σε μορφή **pdf** με:

- *Σχήμα του αυτομάτου πεπερασμένων καταστάσεων (FA) που αντιστοιχεί στη ζητούμενη προδιαγραφή. Σημειώστε ποιες είναι οι καταστάσεις αποδοχής. Οποιαδήποτε μορφή σχήματος είναι αποδεκτή, ακόμα και φωτογραφία κινητού,* ***αρκεί το σχήμα να φαίνεται καθαρά****.*
- *Αποτελέσματα εξόδου (screenshot) του προγράμματός σας.*
- *Αναφορά σε πηγές που πιθανόν χρησιμοποιήσατε.*



ΠΗΓΕΣ:site του μαθηματος

```
"""
Κώδικας που θα χρησιμοποιηθεί ως βάση για την 1η εργασία
των Μεταγλωττιστών (αναγνώριση κειμένου μέσω αυτομάτου DFA).
ΠΡΟΣΟΧΗ: Προσθήκες στον κώδικα επιτρέπονται μόνο
στα σημεία (Α), (Β) και (Γ) - διαβάστε τα σχόλια!
"""


transitions = {
        's0':{'DIGIT':'s1','.':'s2'},
        's1':{'DIGIT':'s1','.':'s3'},
        's2':{'DIGIT':'s3'},
        's3':{'DIGIT':'s3'}
        # (Α) Συμπληρώστε τον πίνακα μεταβάσεων ως λεξικό (dictionary).
        # Η αρχική κατάσταση πρέπει να ονομάζεται 's0'.
        # Για λεπτομέρειες δείτε στο:
        # http://mixstef.github.io/courses/compilers/lecturedoc/unit1/module1.html#id7
            }


accepts = {
        's3':'FLOAT_TOKEN'
        # (Β) Συμπληρώστε το λεξικό των καταστάσεων αποδοχής και των
        # αντίστοιχων επιστρεφόμενων συμβόλων (tokens)
        # Για λεπτομέρειες δείτε στο:
        # http://mixstef.github.io/courses/compilers/lecturedoc/unit1/module1.html#id8
          }


def get_char(text,pos):
        """ Returns char (or char category) at position `pos` of `text`,
        or None if out of bounds. """

        if pos<0 or pos>=len(text): return None

        c = text[pos]
        if c>='0' and c<='9':
                return 'DIGIT'
        # (Γ) Προαιρετικά, μπορείτε να ομαδοποιήσετε τους
        # χαρακτήρες εισόδου εδώ.
        # Για λεπτομέρειες δείτε στο:
        # http://mixstef.github.io/courses/compilers/lecturedoc/unit1/module1.html#id1
        return c

# Δεν επιτρέπεται η παρέμβαση στον κώδικα από αυτό το σημείο και κάτω!

def scan(text,transitions,accepts,state):
        """ Starting from inital `state`, scans `text` while transitions
        exist in `transitions` dict. After that, if on a state belonging to
        `accepts` dict, returns the corresponding token object, else None.
```

```
def scan(text,transitions,accepts,state):
        """ Starting from inital `state`, scans `text` while transitions
        exist in `transitions` dict. After that, if on a state belonging to
        `accepts` dict, returns the corresponding token object, else None.
        """

        # initial position on text
        pos = 0

        # memory object for last seen accepting state
        matched = None

        while True:
                c = get_char(text,pos)  # get next char (or char category)

                if state in transitions and c in transitions[state]:
                        state = transitions[state][c]   # set new state
                        pos += 1        # advance to next char

                        # remember if current state is accepting
                        if state in accepts:
                                matched = { 'token':  accepts[state],
                                            'lexeme': text[:pos] }

                else:   # no transition found, return last match or None
                        return matched


# testing inputs
for test in ['12.456','6789.','.66998','1234','.']:
        m = scan(test,transitions,accepts,'s0')
        print("Testing '{}'\nResult: {}\n".format(test,m))
```

```
П2017153:~$ cd Downloads
П2017153:~/Downloads$ ls
dfa.py
П2017153:~/Downloads$ nano dfa.py
П2017153:~/Downloads$ python3 dfa.py
Testing '12.456'
Result: {'token': 'FLOAT_TOKEN', 'lexeme': '12.456'}

Testing '6789.'
Result: {'token': 'FLOAT_TOKEN', 'lexeme': '6789.'}

Testing '.66998'
Result: {'token': 'FLOAT_TOKEN', 'lexeme': '.66998'}

Testing '1234'
Result: None

Testing '.'
Result: None
```