ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ
ΘΕΩΡΙΑ ΥΠΟΛΟΓΙΣΜΟΥ
ΘΕΩΡΙΑ

ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΟ ΠΕΠΕΡΑΣΜΕΝΟ ΑΥΤΟΜΑΤΟ

ΘΕΩΡΙΑ ΥΠΟΛΟΓΙΣΜΟΥ

Ον\Επ :Χρήστος Χούθης
ΑΜ : 161094

Τεκμηρίωση Εργασίας

Αρχικά θα κατασκευάσουμε ένα πρόγραμμα γραμμένο σε **γλώσσα C.** Το πρόγραμμα αφορά την υλοποίηση ενός Ντετερμινιστικού Πεπερασμένου Αυτόματου το οποίο θα διαβάζουμε από ένα αρχείο με όνομα “dfa.txt”. Ο χρήστης ύστερα θα δίνει διάφορες λέξεις και το πρόγραμμα θα εμφανίζει αν οι λέξεις είναι αποδεκτές η όχι. Αν οι λέξεις που δίνει ο χρήστης είναι εκτός τις αλφάβητου του ντετερμινιστικού απορρίπτονται από την αρχή.

Ο χρήστης **θα πρέπει** να παρέμβει **στην ακόλουθη γραμμή κώδικα** ώστε να δώσει το **δικό του path** όπου και από εκεί θα διαβάζεται το αρχείο εισόδου:

```
fp = fopen("C:\\Users\\Christos\\Desktop\\dfa.txt","r");
```

Μέσα στις παρενθέσεις ο χρήστης **θα πρέπει να προσδιορίζει το δικό του path** και ύστερα μπορεί να εκτελέσει τον κώδικα (**Το path θα πρέπει να έχει την μορφή του παραπάνω αποτελούμενο από διπλά backslash \\ κάθε φορά**).

Παρακάτω θα επεξηγήσουμε βασικά κομμάτια του κώδικα:

Συνάρτηση CheckString():

Αρχικά **δηλώνουμε την συνάρτηση** ώστε να μπορούμε να την χρησιμοποιήσουμε στο πρόγραμμα.

```
// Συνάρτηση για τον έλεγχο της εισόδου του χρήστη
int CheckString(char *string,char *alphabet);
```

Παρακάτω βλέπουμε την συνάρτηση:

```
/*Συνάρτηση με την οποία κοιτάζουμε αν η είσοδος του χρήστη
είναι σωστή με βάση το αλφάβητο του dfa */
int CheckString(char *string,char *alphabet)
{
    int i = 0, j = 0; //Counter πινάκων
    int flag = 0; // flag ώστε να τσεκάρουμε αν το string εμπεριέχεται στο αλφάβητο
    int s1 = strlen(string); // Μήκος Εισόδου
    int s2 = strlen(alphabet); // Μήκος Αλφαβήτου

    //Τρέχουμε ένα loop για κάθε γράμμα της εισόδου
    for(i=0;i<s1-1;i++)
    {
        flag = 0; //Αρχικοποιούμε με 0 κάθε φορά
        // Για κάθε γράμμα εισόδου ελέγχουμε όλα τα γράμματα τις αλφάβητου
        for(j=0; j<s2; j++)
        {
            if(string[i] == alphabet[j]) //Εάν το γράμμα εισόδου εμπεριέχεται στο αλφάβητο
            flag = 1; //Αρχικοποιούμε το flag με 1
        }

        if (flag == 0) //Εάν το γράμμα εισόδου ΔΕΝ εμπεριέχεται στο αλφάβητο
        return 0; //Επιστρέφουμε 0 και τερματίζουμε την συνάρτηση
    }

    return 1; //Τέλος επιστρέφουμε 1 η συμβολοσειρά εισόδου είναι σωστή
}
```

Περιγραφή Συνάρτησης: Η συνάρτηση δέχεται ως **1ο όρισμα** ένα πίνακα τύπου **'char'** και είναι η είσοδος του χρήστη την οποία θα ελέγξουμε, ως **2ο όρισμα** δέχεται ένα πίνακα τύπου **'char'** και είναι το αλφάβητο το οποίο θα χρησιμοποιήσουμε για να βεβαιώσουμε ότι η είσοδος του χρήστη είναι αποδεκτή.

Λειτουργία Συνάρτησης: Αφού η συνάρτηση δεχτεί και τις 2 παραμέτρους, θα αρχικοποιήσουμε κάποιες απαραίτητες μεταβλητές και θα κρατήσουμε τα μεγέθη των παραμέτρων που δώσαμε. Το **1ο loop** το χρησιμοποιούμε ώστε να ελέγχουμε κάθε φορά ξεχωριστά τα στοιχεία του πίνακα εισόδου και αρχικοποιούμε το **flag = 0** ώστε να ελέγχουμε κάθε γράμμα. Το **2ο loop** το χρησιμοποιούμε ώστε να ελέγχουμε κάθε γράμμα εισόδου εάν υπάρχει στην αλφάβητο, οπότε στην περίπτωση που υπάρχει το γράμμα αρχικοποιούμε το **flag = 1**. **Βγαίνοντας από το 2ο loop** κοιτάζουμε αν το **flag == 0** που αυτό σημαίνει ότι το γράμμα δεν βρέθηκε και **επιστρέφουμε 0** και τερματίζουμε την συνάρτηση. Στην περίπτωση που τα **2 loop τερματίσουν** χωρίς να έχουν βρει κάποιο λάθος **επιστρέφουμε 1** που σημαίνει ότι **η συμβολοσειρά εισόδου εμπεριέχεται στην αλφάβητο και είναι σωστή**.

Αρχικά δίνουμε το **σωστό Path** και ανοίγουμε το αρχείο για **“r” διάβασμα**, έπειτα ελέγχουμε αν η **μεταβλητή 'fp' είναι κενή** που σημαίνει πως **υπάρχει πρόβλημα κατά το άνοιγμα του αρχείου και σε αυτήν την περίπτωση τερματίζουμε το πρόγραμμα**(αλλιώς συνεχίζουμε κανονικά):

```
// Άνοιγμα αρχείου
fp = fopen("C:\\Users\\Christos\\Desktop\\dfa.txt","r");

//Έλεγχος για τυχόν πρόβλημα κατά το άνοιγμα του αρχείου
if(fp == NULL)
{
    perror("Error opening file");
    return -1;
}
```

Ελέγχουμε εάν βρισκόμαστε στην **1 γραμμή του αρχείου** που διαβάζουμε. Έπειτα **μετατρέπουμε το στοιχείο** που διαβάσαμε **σε integer** και το αποθηκεύουμε σε μια μεταβλητή. Η μεταβλητή αυτή**('states')** περιέχει τις καταστάσεις του dfa επομένως ελέγχουμε **εάν οι καταστάσεις είναι >=10 τότε τερματίζουμε το πρόγραμμα**.

```
if(i == 1) //Διάβασμα 1ης γραμμής - Καταστάσεις αυτόματου
{
    states = atoi(line); // Μετατροπή char στοιχείο σε integer
    if(states >= 10) // Έλεγχος για το σύνολο των καταστάσεων
    {
        printf("The automatic must have under ten(<10) states!");
        return 1;
    }
}
```

Ελέγχουμε εάν βρισκόμαστε στην **2 γραμμή του αρχείου** που διαβάζουμε. Αρχικοποιούμε κάποιες βασικές μεταβλητές. Διατρέχουμε τον πίνακα με τα στοιχεία που διαβάσαμε από την 2η γραμμή(πίνακας **line**) και αυξάνουμε δύο μετρητές ώστε να γνωρίζουμε το πλήθος της αλφάβητου (**cnt** – Τον χρησιμοποιούμε αποκλειστικά για το πλήθος των γραμμάτων τις αλφαβήτου, fl_counter – Τον χρησιμοποιούμε για το πλήθος των γραμμάτων τις αλφαβήτου αλλά και για επιπλέον πράξεις). Έπειτα **δεσμεύουμε χώρο δυναμικά** για τον πίνακα που θα περιέχει τα **γράμματα τις αλφάβητου**(Πίνακας **alphabet).** Ύστερα ξαναδιατρέχουμε τον **πίνακα line** ώστε να **εκχωρήσουμε** στον **πίνακα alphabet** τα στοιχεία της **αλφάβητου.** Έπειτα ελέγχουμε εάν σε περίπτωση το πλήθος συμβόλων είναι μεγαλύτερο ή ίσο του 10 (**cnt >= 10**) τερματίζουμε το πρόγραμμα.
Ύστερα στη μεταβλητή **trans** αποθηκεύουμε το σύνολο των καταστάσεων του dfa κάνοντας το πολλαπλασιασμό **states * fl_counter**
(states = Καταστάσεις του dfa & fl_counter = Αριθμός συμβόλων της αλφάβητου).
Έπειτα **δεσμεύουμε δυναμικά χώρο** για τους πίνακες **cs,tr,nw(cs = Τρέχουσα κατάσταση** 1η στήλη, **tr = Τιμή μετάβασης** 2η στήλη, **nw = Νέα κατάσταση** 3η στήλη) με **βάση το σύνολο καταστάσεων**(trans).

```
if(i == 2) //Διάβασμα 2ης γραμμής - Αλφάβητο αυτόματου
{
    int cnt = 0; //Αρχικοποίηση μετρητή
    int j = 0; //Αρχικοποίηση μεταβλητής για προσπέλαση του πίνακα

    // Όσο διαφορετικό της αλλαγής γραμμής('\n') τρέχουμε τον πίνακα
    while(line[j] != '\n')
    {
        if(line[j] != ' ') //Αν το στοιχείο διαφορετικό του κενού(' ')
        {
            cnt++; //Αυξάνουμε τον μετρητή
            fl_counter++; //Counter για το υπολογισμό των γραμμάτων της αλφαβητου
        }
        j++; // Αύξηση για προσπέλαση επόμενου στοιχείου του πίνακα
    }

    alphabet = (char *)malloc(cnt * sizeof(char)); //Δεσμεύουμε δυναμικά χώρο για την αλφάβητο

    j=0; cnt = 0;
    // Όσο διαφορετικό της αλλαγής γραμμής('\n') τρέχουμε τον πίνακα
    while(line[j] != '\n')
    {
        if(line[j] != ' ') //Αν το στοιχείο διαφορετικό του κενού(' ')
            alphabet[cnt++] = line[j]; // Εκχωρούμε τα στοιχεία της αλφαβήτου στον πίνακα
        j++; // Αύξηση για προσπέλαση επόμενου στοιχείου του πίνακα
    }

    if(cnt >= 10) // Έλεγχος για το πλήθος συμβόλων
    {
        printf("The alphabet must have under ten(<10) different symbols!");
        return 2;
    }
    trans = states * fl_counter; // Σύνολο των καταστάσεων
    cs = (char *)malloc(trans * sizeof(char)); //Δεσμεύουμε δυναμικά χώρο για τις τρέχουσες καταστάσεις (1η στήλη)
    tr = (char *)malloc(trans * sizeof(char)); //Δεσμεύουμε δυναμικά χώρο για τις τιμές μετάβασεις (2η στήλη)
    nw = (char *)malloc(trans * sizeof(char)); //Δεσμεύουμε δυναμικά χώρο για τις νέες καταστάσεις (3η στήλη)
}
```

Ελέγχουμε εάν βρισκόμαστε στην **3 γραμμή του αρχείου** που διαβάζουμε. Τότε στην προκειμένη περίπτωση **εκχωρούμε στην μεταβλητή start το 1ο στοιχείο που βρίσκεται πίνακα line** όπου και είναι η **αρχική κατάσταση**.

```
if(i==3) //Διάβασμα 3ης γραμμής - Αρχική κατάσταση
    start = line[0]; // Αποθηκέυουμε την αρχική κατάσταση
```

Ελέγχουμε εάν βρισκόμαστε στην **4 γραμμή του αρχείου** που διαβάζουμε. Αρχικοποιούμε έναν μετρητή(cnt) ώστε να μετρήσουμε το πλήθος στοιχείων της γραμμής, όπως και μια μεταβλητή(j) για προσπέλαση του πίνακα. Χρησιμοποιούμε το **1ο while** ώστε να **βρούμε το πλήθος των στοιχείων της τρέχουσας γραμμής**. **Με βάση το πλήθος** που έχουμε κρατήσει στην μεταβλητή 'cnt' **δεσμεύουμε δυναμικά τον πίνακα 'final' τύπου 'char'**. Στην **2η while εκχωρούμε τα στοιχεία της 4η γραμμής στον πίνακα 'final'** όπου θα είναι ο πίνακας που θα περιέχει τις τελικές καταστάσεις.

```
if(i==4) //Διάβασμα 4ης γραμμής - Τελικές καταστάσεις
{

    int cnt = 0; //Αρχικοποίηση μετρητή
    int j = 0; //Αρχικοποίηση μεταβλητής για προσπέλαση του πίνακα

    // Όσο διαφορετικό της αλλαγής γραμμής('\n') τρέχουμε τον πίνακα
    while(line[j] != '\n')
    {
        if(line[j] != ' ')
        { cnt++; } //Αυξάνουμε τον μετρητή

        j++; // Αύξηση για προσπέλαση επόμενου στοιχείου του πίνακα
    }

    // Δεσμεύουμε μνήμη δυναμικά με βάση το πλήθος των τελικών καταστάσεων
    final = (char *)malloc(cnt * sizeof(char));

    int k=0; j=0; //Αρχικοποιήση μεταβλητών

    // Όσο διαφορετικό της αλλαγής γραμμής('\n') τρέχουμε τον πίνακα
    while(line[j] != '\n')
    {
        if(line[j] != ' ')
        { final[k++] = line[j]; } //Εκχωρούμε σε πίνακα τις τελικές καταστάσεις

        j++; // Αύξηση για προσπέλαση επόμενου στοιχείου του πίνακα
    }
}
```

Ελέγχουμε εάν η γραμμή που βρισκόμαστε είναι μεγαλύτερη του 4 **'if(i>4)'** αυτό σημαίνει ότι από την **5 γραμμή και για όσες γραμμές χρειαστεί** ξεκινάμε να διαβάζουμε τις καταστάσεις που βρίσκονται στο αρχείο 'dfa.txt'. Χρησιμοποιώντας την while ξεκινάμε και **για κάθε γραμμή που διαβάζουμε,** κρατάμε στον **πίνακα 'cs' το 1ο στοιχείο της γραμμής** όπου και είναι το στοιχείο για την τρέχουσα κατάσταση, κρατάμε στον **πίνακα 'tr' το 2ο στοιχείο της γραμμής** όπου και είναι το στοιχείο για την τιμή μετάβασης, κρατάμε στον **πίνακα 'nw' το 3ο στοιχείο της γραμμής** όπου και είναι το στοιχείο για την νέα κατάσταση. Το μοτίβο αυτό επαναλαμβάνεται για όσες γραμμές χρειαστεί και κάθε φορά εκχωρούμε στους πίνακες(cs,tr,nw) τα στοιχεία που χρειαζόμαστε, συγκεκριμένα όταν φτάσουμε στο **'if(trans+4 == 1 && j ==4)'** και γίνει **αληθής η συνθήκη τότε κάνουμε break** και συνεχίζουμε το πρόγραμμα.

```
// Στον παρακάτω έλεγχο διαβάζουμε όλες τις γραμμές απο την 4η γραμμή και μετά,
// και αποθηκεύουμε τα στοιχεία αντίστοιχα στους 3 πίνακες(cs,tr,nw)
if(i > 4)
{
    int j = 0;
    while(line[j] != '\n')
    {
        if(line[j] != ' ') //Αν το στοιχείο διαφορετικό του κενού(' ')
        {
            if(j == 0) //Εάν βρισκόμαστε στο 1ο στοιχείο
                cs[k] = line[j]; //Αποθηκέυουμε το στοιχείο για τις τρέχουσες καταστάσεις
            if(j == 2) //Εάν βρισκόμαστε στο 2ο στοιχείο
                tr[k] = line[j]; //Αποθηκέυουμε το στοιχείο για τις τιμές μετάβασεις
            if(j == 4) //Εάν βρισκόμαστε στο 3ο στοιχείο
                nw[k] = line[j]; //Αποθηκέυουμε το στοιχείο για τις νέες καταστάστεις
        }

            //Εάν βρεθούμε στην τελευταία γραμμή του αρχείο και τελιώσουμε την εκχώρηση κάνουμε break
            if(trans+4 == i && j == 4)
            break;

        j++; // Αύξηση για προσπέλαση επόμενου στοιχείου του πίνακα line
    }

    k++; //Αύξηση μετρήτη για τους πίνακες cs,tr,nw ώστε να αυξάνονται παράλληλα
}
```

Ο χρήστης μας δίνει την είσοδο και χρησιμοποιούμε την συνάρτηση fgets με ορίσματα **input = Πίνακας χαρακτήρων** , **30 μέγιστος αριθμός χαρακτήρων**, **stdin = είσοδος από πληκτρολόγιο**.

```
//Διαβάζουμε την λέξη εισόδου απο τον χρήστη
printf("Give the input:");
fgets(input,30,stdin);
```

Χρησιμοποιούμε την συνάρτηση **'CheckString'** ώστε να δούμε αν η είσοδος του χρήστη εμπεριέχεται στο αλφάβητο, **εάν λοιπόν η είσοδος δεν εμπεριέχεται στο αλφάβητο τότε η συνάρτηση επιστρέφει '0'** και τερματίζουμε το πρόγραμμα (αλλιώς συνεχίζουμε την εκτέλεση του προγράμματος).

```
// Ελέγχουμε αν η συμβολοσειρά εισόδου είναι αποδεκτή
// σύμφωνα με το αλφάβητο
if(CheckString(input,alphabet) == 0)
  {
      printf("The input is wrong!");
      return 1;
  }
```

Στην μεταβλητή **'length_in' κρατάμε το μέγεθος της εισόδου** γιατί θα μας χρειαστεί παρακάτω. **Αρχικοποιούμε** την **μεταβλητή 'final_state'** με την **αρχική κατάσταση του dfa**. Αρχικοποιούμε την μεταβλητή 'flag' την οποία θα χρησιμοποιήσουμε για παρακάτω έλεγχο.

```
int length_in = strlen(input); //Μήκος εισόδου
char final_state = start; // Αρχικοποιούμε την τελική κατάσταση με την αρχική κατάσταση
int flag = 0; // flag για έλεγχο
```

Τρέχουμε το **1ο loop** για **κάθε χαρακτήρα της εισόδου** αρχικοποιώντας κάθε φορά 2 βασικές μεταβλητές. Τρέχουμε το **2ο loop** για **όσες είναι και οι καταστάσεις του dfa**, κάθε φορά **ελέγχουμε εάν η μεταβλητή 'final_state' είναι ίση με το στοιχείο της 'cs[j]'** (δηλαδή αν η τρέχουσα κατάσταση του χρήστη είναι ίση με την τρέχουσα κατάσταση του πίνακα που έχουμε αποθηκεύσει τις τρέχουσες καταστάσεις) **ΚΑΙ εάν** ο **τρέχον χαρακτήρας του χρήστη('input[i]')** είναι **ίσος** με **τιμή μετάβασης 'tr[j]' ΤΟΤΕ** η **μεταβλητή 'final_state' παίρνει την νέα κατάσταση** από τον **πίνακα 'nw[j]'** όπου και εμπεριέχονται η νέες καταστάσεις. Ύστερα κάνουμε το 'flag=1' αφού έχουμε βρει την κατάσταση μετάβασης και συνεχίζουμε με την επόμενη.

ΣΥΝΟΠΤΙΚΑ:

- Τρέχουμε το **1ο loop** για κάθε χαρακτήρα εισόδου του χρήστη
- Τρέχουμε το **2ο loop**, **Εάν βρούμε τα κατάλληλα στοιχεία που χρειαζόμαστε ΤΟΤΕ εκχωρούμε στη μεταβλητή 'final_state' την νέα κατάσταση**
- **Εάν** βρούμε την **νέα κατάσταση κάνουμε break** ώστε να πάμε στην επόμενη
- Πάμε **ξανά στο 1ο loop** για τον επόμενο χαρακτήρα και **επαναλαμβάνουμε**

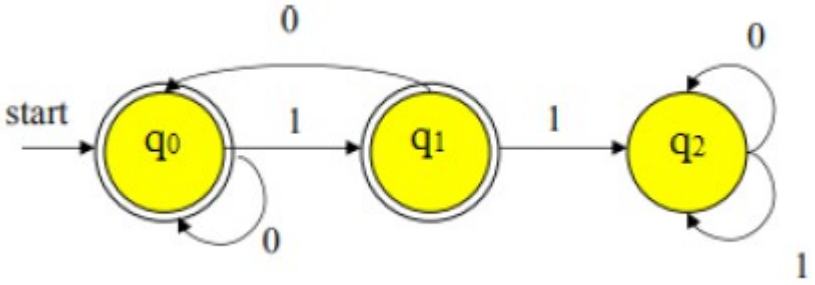
```
for(i=0;i<length_in-1;i++)
{
    //Αρχικοποιήση μεταβλητών
    int j = 0;
    flag = 0;

    /* Τρέχουμε ένα loop με βάση το σύνολο των καταστάσεων, κάθε φορά που
    βρίσκουμαι την κατάσταση που χρειαζόμαστε για να γίνει η μετάβαση
    ορίζουμε το flag = 1 ώστε μετά το τον έλεγχο, να κάνουμε έναν
    έλεγχο αν το flag == 1 και τότε κάνουμε break αφού έχουμε ήδη βρει την
    κατάσταση που χρειαζόμαστε και συνεχίζουμε με το επόμενο γράμμα εισόδου */
    for(j=0;j<trans;j++)
    {
        if(final_state == cs[j] && input[i] == tr[j])
        {
            //printf("final_state: %c == cs[j]: %c\n",final_state,cs[j]);
            //printf("input:       %c == tr[j]: %c\n",input[i],tr[j]);
            printf("INPUT -> %c\nCURRENT_STATE -> q%c\n",input[i],final_state);
            final_state = nw[j];
            printf("NEW_STATE     -> 'q%c'\n\n",final_state);
            flag = 1;
        }

        if(flag==1)
        break;
    }
}
```

Αφού πλέον στην **μεταβλητή 'final_state' εμπεριέχεται η τελική κατάσταση**, χρησιμοποιούμε το **loop ώστε να κοιτάξουμε αν η τελική κατάσταση** με βάση την είσοδο του χρήστη **βρίσκεται στον πίνακα 'final'** όπου εκεί έχουμε κρατήσει της αποδεκτές τελικές καταστάσεις. Εάν ισχύει κάνουμε το **'flag=1'**.
Παρακάτω ελέγχουμε **εάν το 'flag == 1'** τότε η είσοδος είναι **αποδεκτή**. **Αλλιώς** η είσοδος **δεν είναι αποδεκτή**.

```
for(i=0;i<strlen(final);i++)
{
    if(final_state == final[i])
    flag = 1;
}

//Εαν είναι αποδεκτή κατάσταση
if(flag == 1)
{
    printf("----------------------------");
    printf("\nThe Input is acceptable!\n");
    printf("The Final State is q%c\n",final_state);
    printf("----------------------------");
}
else
{
    printf("------------------------------");
    printf("\nThe Input is NOT acceptable!\n");
    printf("The Final State is q%c\n",final_state);
    printf("------------------------------");
}
```

Παρακάτω θα δούμε παραδείγματα για συγκεκριμένα ντετερμινιστικά πεπερασμένα αυτόματα:

1) Αρχικά θα τρέξουμε το παράδειγμα της άσκησης, παρακάτω βλέπουμε το σχήμα:



Αρχείο εισόδου:
3
0 1
0
0 1
0 1 1
0 0 0
1 1 2
1 0 0
2 1 2
2 0 2

Παρακάτω βλέπουμε κάποια παραδείγματα για εισόδους και εξόδους του προγράμματος:

Είσοδος: 0101
Έξοδος:

```
Give the input:0101
INPUT -> 0
CURRENT_STATE -> q0
NEW_STATE     -> 'q0'

INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

INPUT -> 0
CURRENT_STATE -> q1
NEW_STATE     -> 'q0'

INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

--------------------------
The Input is acceptable!
The Final State is q1
--------------------------
```

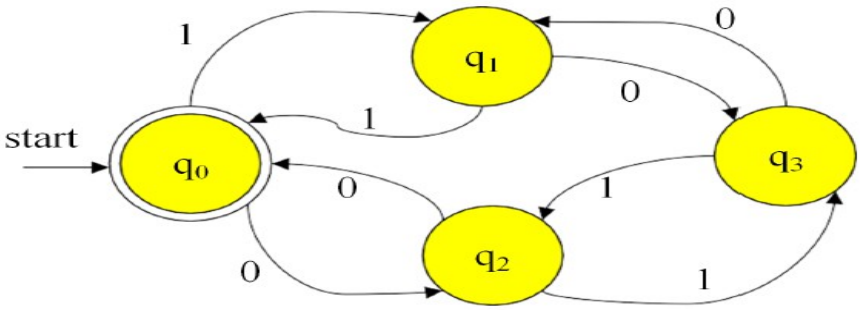
Είσοδος: 11010
Έξοδος:

```
Give the input:11010
INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

INPUT -> 1
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> 0
CURRENT_STATE -> q2
NEW_STATE     -> 'q2'

INPUT -> 1
CURRENT_STATE -> q2
NEW_STATE     -> 'q2'

INPUT -> 0
CURRENT_STATE -> q2
NEW_STATE     -> 'q2'

------------------------------
The Input is NOT acceptable!
The Final State is q2
------------------------------
```

Είσοδος: 1001
Έξοδος:

```
Give the input:1001
INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

INPUT -> 0
CURRENT_STATE -> q1
NEW_STATE     -> 'q0'

INPUT -> 0
CURRENT_STATE -> q0
NEW_STATE     -> 'q0'

INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

--------------------------
The Input is acceptable!
The Final State is q1
--------------------------
```

2) Ένα διαφορετικό ντετερμινιστικό πεπερασμένο αυτόματο:



Αρχείο εισόδου:

4
0 1
0
0
0 0 2
0 1 1
1 0 3
1 1 0
2 0 0
2 1 3
3 0 1
3 1 2

Παρακάτω βλέπουμε κάποια παραδείγματα για εισόδους και εξόδους του προγράμματος:

Είσοδος: 0101
Έξοδος:

```
Give the input:0101
INPUT -> 0
CURRENT_STATE -> q0
NEW_STATE     -> 'q2'

INPUT -> 1
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

INPUT -> 0
CURRENT_STATE -> q3
NEW_STATE     -> 'q1'

INPUT -> 1
CURRENT_STATE -> q1
NEW_STATE     -> 'q0'

--------------------------
The Input is acceptable!
The Final State is q0
--------------------------
```

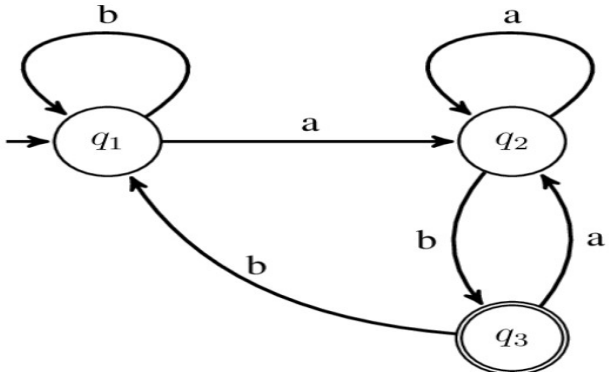
Είσοδος: 1100
Έξοδος:

```
Give the input:1100
INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE    -> 'q1'

INPUT -> 1
CURRENT_STATE -> q1
NEW_STATE    -> 'q0'

INPUT -> 0
CURRENT_STATE -> q0
NEW_STATE    -> 'q2'

INPUT -> 0
CURRENT_STATE -> q2
NEW_STATE    -> 'q0'

--------------------------
The Input is acceptable!
The Final State is q0
--------------------------
```

Είσοδος: 10111
Έξοδος:

```
Give the input:10111
INPUT -> 1
CURRENT_STATE -> q0
NEW_STATE    -> 'q1'

INPUT -> 0
CURRENT_STATE -> q1
NEW_STATE    -> 'q3'

INPUT -> 1
CURRENT_STATE -> q3
NEW_STATE    -> 'q2'

INPUT -> 1
CURRENT_STATE -> q2
NEW_STATE    -> 'q3'

INPUT -> 1
CURRENT_STATE -> q3
NEW_STATE    -> 'q2'

------------------------------
The Input is NOT acceptable!
The Final State is q2
------------------------------
```

3) Άλλο ένα διαφορετικό ντετερμινιστικό πεπερασμένο αυτόματο:



Αρχείο εισόδου:

3
a b
1
3
1 a 2
1 b 1
2 a 2
2 b 3
3 a 2
3 b 1

Παρακάτω βλέπουμε κάποια παραδείγματα για εισόδους και εξόδους του προγράμματος:

Είσοδος: babb
Έξοδος:

```
Give the input:babb
INPUT -> b
CURRENT_STATE -> q1
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

INPUT -> b
CURRENT_STATE -> q3
NEW_STATE     -> 'q1'

------------------------------
The Input is NOT acceptable!
The Final State is q1
------------------------------
```

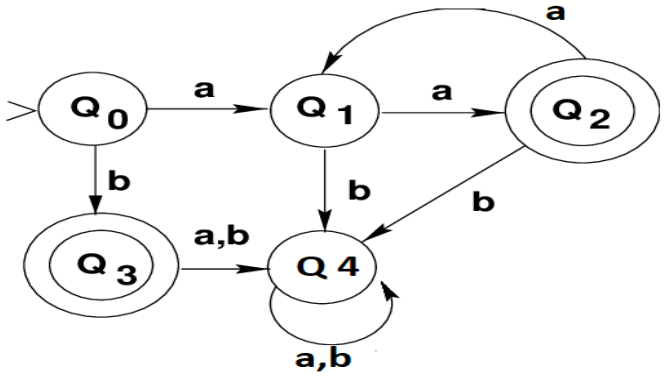
Είσοδος: aabab
Έξοδος:

```
Give the input:aabab
INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> a
CURRENT_STATE -> q2
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

INPUT -> a
CURRENT_STATE -> q3
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

--------------------------
The Input is acceptable!
The Final State is q3
--------------------------
```

Είσοδος: abbbaaba
Έξοδος:

```
Give the input:abbbaaba
INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

INPUT -> b
CURRENT_STATE -> q3
NEW_STATE     -> 'q1'

INPUT -> b
CURRENT_STATE -> q1
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> a
CURRENT_STATE -> q2
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q3'

INPUT -> a
CURRENT_STATE -> q3
NEW_STATE     -> 'q2'

------------------------------
The Input is NOT acceptable!
The Final State is q2
------------------------------
```

4) Άλλο ένα διαφορετικό ντετερμινιστικό πεπερασμένο αυτόματο:



Αρχείο εισόδου:

5
a b
0
2 3
0 a 1
0 b 3
1 a 2
1 b 4
2 a 1
2 b 4
3 a 4
3 b 4
4 a 4
4 b 4

Παρακάτω βλέπουμε κάποια παραδείγματα για εισόδους και εξόδους του προγράμματος:

Είσοδος: aaaab
Έξοδος:

```
Give the input:aaaab
INPUT -> a
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> a
CURRENT_STATE -> q2
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> b
CURRENT_STATE -> q2
NEW_STATE     -> 'q4'

------------------------------
The Input is NOT acceptable!
The Final State is q4
------------------------------
```

Είσοδος: baba
Έξοδος:

```
Give the input:baba
INPUT -> b
CURRENT_STATE -> q0
NEW_STATE     -> 'q3'

INPUT -> a
CURRENT_STATE -> q3
NEW_STATE     -> 'q4'

INPUT -> b
CURRENT_STATE -> q4
NEW_STATE     -> 'q4'

INPUT -> a
CURRENT_STATE -> q4
NEW_STATE     -> 'q4'

------------------------------
The Input is NOT acceptable!
The Final State is q4
------------------------------
```

Είσοδος: aaaa
Έξοδος:

```
Give the input:aaaa
INPUT -> a
CURRENT_STATE -> q0
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

INPUT -> a
CURRENT_STATE -> q2
NEW_STATE     -> 'q1'

INPUT -> a
CURRENT_STATE -> q1
NEW_STATE     -> 'q2'

--------------------------
The Input is acceptable!
The Final State is q2
--------------------------
```