

# Laboratorio di Linguaggi Formali e Traduttori

## LFT LAB T1/T3

Viviana Patti  
Corso di Studi in Informatica  
a.a. 2022/2023

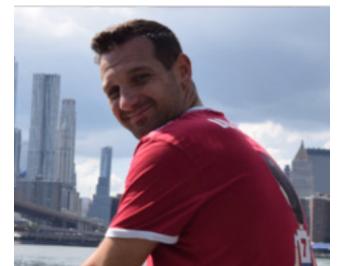
# LFT Lab - 3 CFU



**Docente: Viviana Patti**  
Dipartimento di Informatica  
Università di Torino  
E-mail: [patti@di.unito.it](mailto:patti@di.unito.it)  
Tel.: 011 6706804  
Homepage:  
<https://www.unito.it/persone/vpatti>



- **Ufficio:** Dipartimento di Informatica (terzo piano - ufficio 6)
- **Orario di ricevimento:** su appuntamento ogni mercoledì dalle 17 alle 18
- Stanza virtuale per il ricevimento online (su richiesta):  
<https://unito.webex.com/meet/viviana.patti> | 843118936
- Assistente per LFT lab: Giovanni (detto Gianni) Forlastro: [gianni.forlastro@gmail.com](mailto:gianni.forlastro@gmail.com) .
- Pagina del corso su Piattaforma Moodle:
- LFT LAB T1: <https://informatica.i-learn.unito.it/course/view.php?id=2582>
- LFT LAB T3: <https://informatica.i-learn.unito.it/course/view.php?id=2584>



# Orario delle lezioni del secondo anno del Corso B

Ora	Lun	Mar	Mer	Gio	Ven
9-10	<a href="#">SO lab T3 Unix (Laboratorio Dijkstra)</a> <a href="#">SO lab T4 Unix (Laboratorio Turing)</a>	<a href="#">LFT B (Aula D)</a>			<a href="#">SO B (Aula D)</a>
10-11	<a href="#">SO lab T3 Unix (Laboratorio Dijkstra)</a> <a href="#">SO lab T4 Unix (Laboratorio Turing)</a>	<a href="#">LFT B (Aula D)</a>			<a href="#">SO B (Aula D)</a>
11-12	<a href="#">LFT lab T3 (Laboratorio Dijkstra)</a> <a href="#">lun 3/10 (Laboratorio Turing)</a>	<a href="#">SO B (Aula D)</a>	<b>Triennale Informatica - Secondo Anno - Corso A 3 - 7 ott 2022</b> <a href="#">SO lab T4 (Laboratorio von Neumann)</a>	<a href="#">SO B (Aula D)</a>	<a href="#">LFT lab T3 (Laboratorio Dijkstra)</a> <a href="#">LFT lab T4 (Laboratorio Turing)</a>
Tutto il giorno		<b>mar 4/10</b>	<b>mer 5/10</b>	<b>gio 6/10</b>	<b>ven 7/10</b>
08	<a href="#">LFT lab T3 (Laboratorio Dijkstra)</a>	<a href="#">SO B (Aula D)</a>		<a href="#">SO B (Aula D)</a>	<a href="#">LFT lab T3 (Laboratorio Dijkstra)</a>
12-13	<a href="#">LFT lab T4 (Laboratorio Turing)</a>				<a href="#">LFT lab T4 (Laboratorio Turing)</a>
09	09:00 - 11:00 <a href="#">SO A - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) D. GUNETTI Aula C (Dipartimento di Informatica) - Cognomi A-K</a>	09:00 - 12:00 <a href="#">SO lab T1 Unix - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) D. RADICIONI, M. DE PIERRO Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi A-K</a>	09:00 - 11:00 <a href="#">EPS A - ELEMENTI DI PROBABILITA' E STATISTICA MFN0600 - INFORMATICA (2) G. D'ONOFRIO, R. SIROVICH Aula C (Dipartimento di Informatica)</a>	09:00 - 11:00 <a href="#">EPS A - ELEMENTI DI PROBABILITA' E STATISTICA MFN0600 - INFORMATICA (2) G. D'ONOFRIO, R. SIROVICH Aula C (Dipartimento di Informatica)</a>	09:00 - 11:00 <a href="#">EPS A - ELEMENTI DI PROBABILITA' E STATISTICA MFN0600 - INFORMATICA (2) G. D'ONOFRIO, R. SIROVICH Aula C (Dipartimento di Informatica)</a>
10	11:00 - 13:00 <a href="#">LFT A - LINGUAGGI FORMALI E TRADUTTORI</a>		11:00 - 13:00 <a href="#">LFT A - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) J. SPROSTON Aula C (Dipartimento di Informatica)</a>	11:00 - 13:00 <a href="#">LFT lab T2 - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) J. SPROSTON Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi A-K</a>	11:00 - 13:00 <a href="#">SO A - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) D. GUNETTI Aula C (Dipartimento di Informatica) - Cognomi A-K</a>
11	14:00 - 16:00 <a href="#">EPS B (Aula D)</a>	13:00 - 16:00 <a href="#">LFT lab T1 - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) V. PATTI Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi A-K</a>	14:00 - 17:00 <a href="#">SO lab T1 C - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) M. DE PIERRO, D. R. MFN0601 - Aula A Turing Informatica (Dipartimento di Infor E. BINI Cognomi A-K</a>	14:00 - 17:00 <a href="#">EPS B (Aula D)</a>	15:00 - 16:00 <a href="#">SO lab T3 C (Laboratorio Dijkstra)</a> <a href="#">SO lab T4 C (Laboratorio Turing)</a>
12	14:00 - 17:00 <a href="#">SO lab T2 Unix - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) E. BINI Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi A-K</a>				<a href="#">SO lab T3 C (Laboratorio Dijkstra)</a> <a href="#">SO lab T4 C (Laboratorio Turing)</a>
13					
14					
15					
16					
17					

# Orario T1

# Orario delle lezioni del secondo anno del Corso A

Ora	Lun	Mar	Mer	Gio	Ven
9-10	<a href="#">LFT A</a> (Aula C)	<a href="#">LFT lab T1</a> (Laboratorio Dijkstra) <a href="#">LFT lab T2</a> (Laboratorio Von Neumann)		<a href="#">LFT lab T1</a> (Laboratorio Dijkstra) <a href="#">LFT lab T2</a> (Laboratorio Von Neumann)	<a href="#">EPS A</a> (Aula C)
10-11	<a href="#">LFT A</a> (Aula C)	<a href="#">LFT lab T1</a> (Laboratorio Dijkstra) <a href="#">LFT lab T2</a> (Laboratorio Von Neumann)		<a href="#">LFT lab T1</a> (Laboratorio Dijkstra) <a href="#">LFT lab T2</a> (Laboratorio Von Neumann)	<a href="#">EPS A</a> (Aula C)

# Orario T3

## Triennale Informatica - Secondo Anno - Corso B

10 – 14 ott 2022

	lun 10/10	mar 11/10	mer 12/10	gio 13/10	ven 14/10
Tutto il giorno					
08					
09	09:00 - 11:00 SO lab T3 Unix - SIS MFN0601 - INFORMATICA (2) E. BINI Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z	09:00 - 11:00 LFT B - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) G. POZZATO Aula D (Dipartimento di Informatica) - Cognomi I-Z	09:00 - 11:00 SO lab T3 C - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) E. BINI Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z	09:00 - 11:00 SO lab T4 C - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) M. DE PIERRO, C. SCHIFANELLA Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z	09:00 - 11:00 SO lab T4 C - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) M. DE PIERRO, C. SCHIFANELLA Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z
10	11:00 - 14:00 LFT lab T3 - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) V. PATTI Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z	11:00 - 13:00 SO B - SISTEMI OPERATIVI MFN0601 - INFORMATICA (2) C. BAROGLIO Aula D (Dipartimento di Informatica) - Cognomi I-Z	11:00 - 13:00 LFT B - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) G. POZZATO Aula D (Dipartimento di Informatica) - Cognomi I-Z	11:00 - 13:00 EPS B - ELEMENTI DI PROBABILITA' E STATISTICA MFN0600 - INFORMATICA (2) R. SIROVICH, M. GIRAUDO Aula D (Dipartimento di Informatica) - Cognomi I-Z	11:00 - 14:00 LFT lab T4 - LINGUAGGI FORMALI E TRADUTTORI MFN0603 - INFORMATICA (2) L. DI CARO Aula E W. Dijkstra Informatica (Dipartimento di Informatica) - Cognomi L-Z
11					
12					
13					
14					
15					
16					
17					
17-18					<a href="#">EPS A</a> (Aula C)
18-19					

# T1,T2,T3 o T4?

- I docenti di teoria sono:
  - Corso A: iniziale tra A e K: Jeremy Sproston
  - Corso B: iniziale tra L e Z: Gianluca Pozzato
- Regole per suddivisione di studenti in turni:
- Turno 1: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è dispari; docente: Viviana Patti
- Turno 2: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è pari; docente: Jeremy Sproston
- Turno 3: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è dispari; docente: Viviana Patti
- Turno 4: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è pari; docente: Luigi Di Caro

# Forum di discussione e supporto on-line al corso

- LFT LAB T1: <https://informatica.i-learn.unito.it/course/view.php?id=2582>
- LFT LAB T3: <https://informatica.i-learn.unito.it/course/view.php?id=2584>
- Sulla piattaforma I-learn sono disponibili **forum di discussione** dedicati per gli argomenti affrontati durante il corso
- Utile per scambiare opinioni tra i vari gruppi di lavoro e con il docente.
- L'iscrizione al forum principale è effettuata automaticamente
  - possibile ‘disiscriversi’ ma sconsigliato prima del superamento dell'esame per poter sempre ricevere in modo tempestivo le comunicazioni

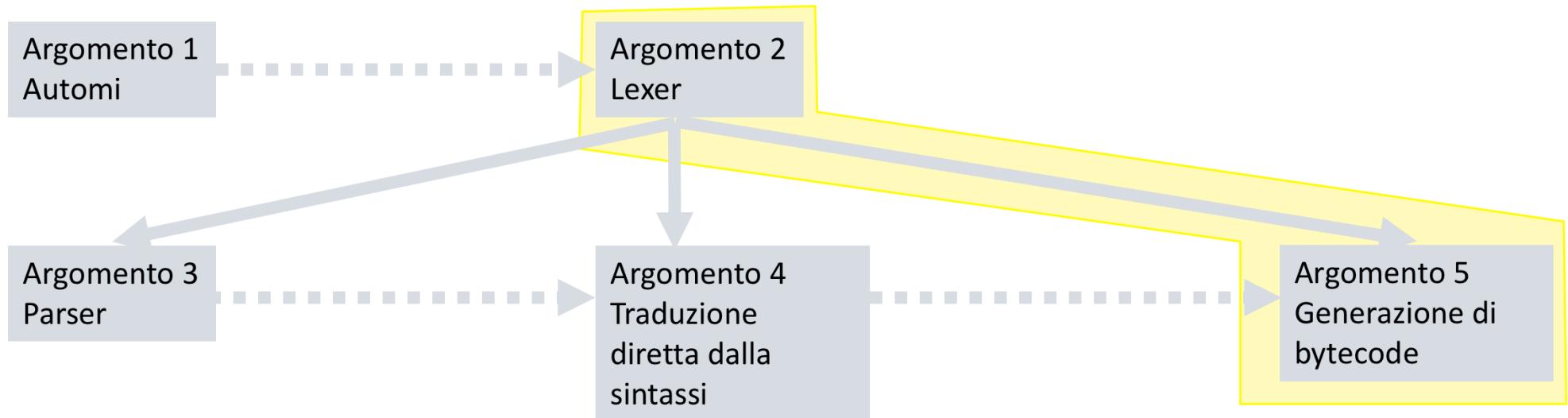
The screenshot shows a Moodle course page for the course 'LFTLabT1-2223'. The header includes the University of Turin logo, the text 'Corsi di Studi in Informatica', and a user profile for 'Patti Viviana'. The navigation bar has links for 'Moodle community', 'Unito', 'HelpDesk', 'I miei corsi', and 'Italiano (it)'. The main content area displays course information: 'Home > I miei corsi > Anno Accademico 22/23 > Secondo anno Laurea DM270 > LFTLabT1-2223'. A sidebar on the left titled 'Navigazione' lists course sections like 'Dashboard', 'Pagine del sito', 'I miei corsi', 'Anno Accademico 22/23', 'Secondo anno Laurea DM270', 'LFTLabT3-2223', 'LFTLabT1-2223', 'Partecipanti', 'Badge', 'Competenze', 'Valutazioni', 'Introduzione', '4 ottobre - 10 ottobre', '11 ottobre - 17 ottobre', and '18 ottobre - 24 ottobre'. Another sidebar on the right titled 'Termina modifica' contains a button 'Aggiungi un blocco' and a dropdown menu 'Aggiungi...'. The central content area lists course details: 'Turno T1: cognomi A-K (corso A), numero matricola dispari', 'Docente: Viviana Patti', 'Data d'inizio: 4 ottobre 2022', 'Ricevimento Prof. Viviana Patti su appuntamento ogni mercoledì dalle 17 alle 18 Ufficio: 032\_A\_P03\_3010, Dipartimento di Informatica, Terzo Piano.', 'Stanza virtuale per il ricevimento: https://unito.webex.com/meet/viviana.patti', and 'Forum di discussione sugli argomenti del corso'.

# Svolgimento e valutazione del progetto di laboratorio



# Progetto di laboratorio LFT LAB

- Il progetto di laboratorio consiste in una serie di **esercitazioni** assistite mirate allo sviluppo di un semplice **traduttore**.
- Il corretto svolgimento di tali esercitazioni presuppone
  - una buona **conoscenza del linguaggio di programmazione Java**
  - una buona conoscenza degli **argomenti di teoria del corso Linguaggi Formali e Traduttori**.



# Descrizione del progetto

- Documento su Moodle che arricchiremo via via
- Progetto di laboratorio  
valido **fino alla sessione  
di febbraio 2024.**

Laboratorio di Linguaggi Formali e Traduttori  
Corso di Studi in Informatica  
A.A. 2022/2023

Luigi Di Caro, Viviana Patti e Jeremy Sproston  
Dipartimento di Informatica — Università degli Studi di Torino

Versione del 4 ottobre 2022

**Sommario**  
Questo documento descrive le esercitazioni di laboratorio e le modalità d'esame del corso di *Linguaggi Formali e Traduttori* per l'A.A. 2022/2023.

**Svolgimento e valutazione del progetto di laboratorio**  
È consigliato sostenere l'esame nella prima sessione d'esame dopo il corso.

**Supporto on-line al corso e forum di discussione**  
Sulla piattaforma i-learn sono disponibili due forum: il primo è dedicato alla pubblicazione di annunci e notizie di carattere generale, mentre il secondo è un forum di discussione dedicato per gli argomenti affrontati durante il corso. L'iscrizione al forum annunci è effettuata automaticamente, è possibile disiscriversi ma è consigliabile farlo solo a seguito del superamento dell'esame per poter sempre ricevere in modo tempestivo le comunicazioni effettuate dal docente.

**Progetto di laboratorio**  
Il progetto di laboratorio consiste in una serie di esercitazioni assistite mirate allo sviluppo di un semplice traduttore. Il corretto svolgimento di tali esercitazioni presupone una buona conoscenza del linguaggio di programmazione Java e degli argomenti di teoria del corso Linguaggi Formali e Traduttori.

**Modalità dell'esame di laboratorio**  
Per sostenere l'esame a un appello è necessario prenotarsi. L'esame di laboratorio è orale e individuale, anche se il codice è stato sviluppato in collaborazione con altri studenti. Durante l'esame vengono accertati: il corretto svolgimento della prova di laboratorio; la comprensione della sua struttura e del suo funzionamento; la comprensione delle parti di teoria correlata al laboratorio stesso.

**Note importanti**

- Per poter discutere il laboratorio è *necessario* aver prima superato la prova scritta relativa al modulo di teoria. L'esame di laboratorio deve essere superato nella sessione d'esame in cui viene superato lo scritto, altrimenti lo scritto deve essere sostenuto nuovamente.

# Modalità dell'esame di laboratorio

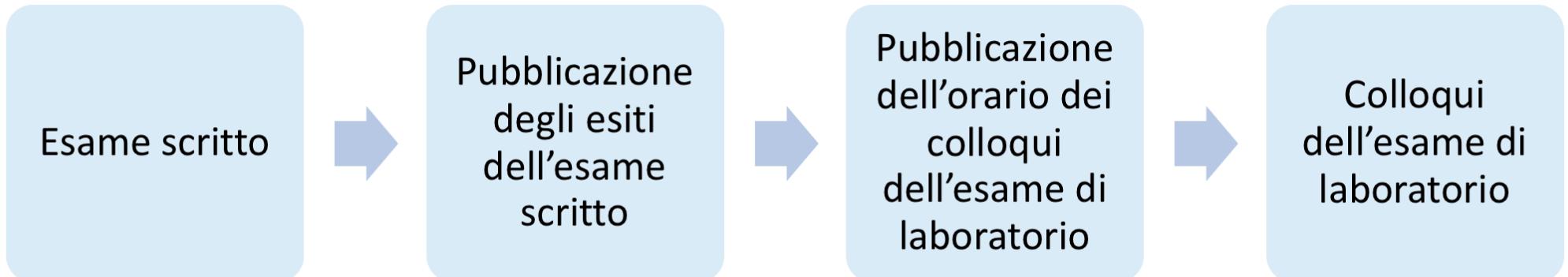
- Per sostenere l'esame a un appello è necessario prenotarsi su MyUnito.
- Ogni studente deve presentarsi all'esame con il codice di **tutti gli esercizi obbligatori presentati durante il corso di laboratorio** (il codice **non** deve essere consegnato al docente prima dell'esame).
- Il progetto di laboratorio può essere svolto individualmente o in gruppi formati da al massimo 3 studenti.
- L'esame di laboratorio consiste in un colloquio **orale** ed è **individuale**, anche se il codice è stato sviluppato in collaborazione con altri studenti.
- Durante l'esame vengono accertati:
  - il **corretto svolgimento** della prova di laboratorio
  - la **comprensione** della sua struttura e del suo **funzionamento**
  - la **comprensione delle parti di teoria** correlata al laboratorio

# Modalità dell'esame di laboratorio

- Per poter discutere il laboratorio è necessario aver **prima** superato la **prova scritta** relativa al modulo di teoria.
- L'esame di laboratorio deve essere superato **nella stessa sessione d'esame** in cui viene superato lo scritto, altrimenti lo scritto deve essere sostenuto nuovamente.
- Le sessioni d'esame in un a.a. sono tre:  
(1) **gennaio/febbraio**, (2) **giugno/luglio**, e (3) **settembre**.
  - (1) gennaio/febbraio  
(scritto -> laboratorio / scritto -> laboratorio),
  - (2) giugno/luglio (scritto -> laboratorio / scritto -> laboratorio)
  - (3) settembre (scritto -> laboratorio).

# Modalità dell'esame di laboratorio

- Utilizzeremo più giorni per fare i colloqui dell'esame di laboratorio: il giorno ufficiale dell'appello e i giorni lavorativi successivi, fino a esaurimento della coda di studenti iscritti
- La suddivisione degli studenti tra i giorni dei colloqui sarà comunicata (via email agli studenti iscritti all'appello) dopo la pubblicazione degli esiti dell'ultimo esame scritto di LFT che precede l'appello di laboratorio
- Nel caso in cui non possiate presentarvi in uno o più giorni lavorativi successivi al giorno ufficiale dell'appello: scrivere al docente prima della pubblicazione degli esiti dell'esame scritto



# Modalità dell'esame di laboratorio

- **WARNING!**
- La presentazione di codice “funzionante” non è condizione sufficiente per il superamento della prova di laboratorio
  - è possibile essere respinti presentando codice funzionante se lo studente dimostra di non avere adeguata familiarità con il codice e i concetti correlati.
- Anche se il codice è stato sviluppato in collaborazione con altri studenti, i punteggi ottenuti dai singoli studenti sono indipendenti.
  - Per esempio, a parità di codice presentato, è possibile che uno studente meriti 30, un altro 25 e un altro ancora sia respinto.
- Dal momento che durante la prova è possibile che venga richiesto di apportare modifiche al codice del progetto, è opportuno presentarsi all'esame con un'adeguata conoscenza del progetto e degli argomenti di teoria correlati.

# Modalità dell'esame di laboratorio

- E' fortemente **consigliato** sostenere l'esame nella prima sessione d'esame dopo il corso.



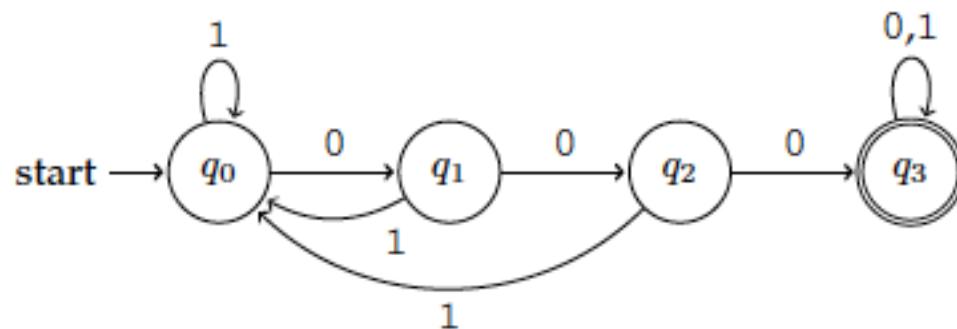
# LFT Calcolo del voto finale

- I voti della prova scritta e della prova di laboratorio sono espressi in trentesimi. Il voto finale è determinato calcolando la media pesata del voto della prova scritta e del laboratorio, secondo il loro contributo in CFU, e cioè

$$\text{voto finale} = \frac{\text{voto dello scritto} \times 2 + \text{voto del laboratorio}}{3} \pm \text{eventuale esito orale}$$

- Con una eventuale modifica nel caso in cui lo studente ha scelto di sostenere una prova orale.
- La prova orale è obbligatoria se lo studente desidera ottenere la lode

# 1. Implementazione di un DFA in Java



# Esercizio 1.1

- Lo scopo di questo esercizio e dei successivi è l’implementazione di un metodo Java che sia in grado di discriminare le stringhe del linguaggio riconosciuto da un automa a stati finiti deterministico (DFA) dato.
- Il primo automa che prendiamo in considerazione, mostrato in Figura 1 del documento, è definito sull’alfabeto  $\{0, 1\}$  e riconosce le stringhe in cui compaiono almeno 3 zeri consecutivi

# Esercizio 1.1 – Progettazione Automa

Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

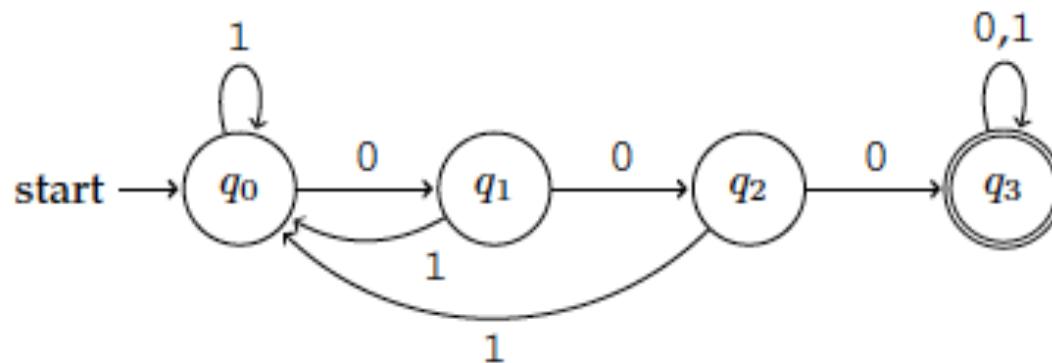


Figura 1: DFA che riconosce stringhe con 3 zeri consecutivi.

# Esercizio 1.1 - Implementazione

## Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

- L'automa è implementato mediante il metodo **scan**:
  - Prende in **input** una **stringa s**
  - Restituisce un valore **booleano** che indica se la stringa appartiene o meno al linguaggio riconosciuto dall'automa.
- Lo **stato dell'automa** è rappresentato per mezzo di una variabile intera **state**
- La variabile **i** contiene l'**indice del prossimo carattere** della stringa s da analizzare.
- Iniziate le variabili

```
public class TreZeri
{
    public static boolean scan(String s)
    {
        int state = 0;
        int i = 0;
```

# Esercizio 1.1 - Implementazione

```
public class TreZeri
{
    public static boolean scan(String s)
    {
        int state = 0;
        int i = 0;

        while (state >= 0 && i < s.length()) {
            final char ch = s.charAt(i++);

            switch (state) {
                case 0:
                    if (ch == '0')
                        state = 1;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 1:
                    if (ch == '0')
                        state = 2;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 2:
                    if (ch == '0')
                        state = 3;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 3:
                    if (ch == '0' || ch == '1')
                        state = 3;
                    else
                        state = -1;
                    break;
            }
        }
        return state == 3;
    }

    public static void main(String[] args)
    {
        System.out.println(scan(args[0]) ? "OK" : "NOPE");
    }
}
```

Il **corpo principale** del metodo è un **ciclo** che, analizzando il contenuto della stringa *s* un carattere alla volta, effettua un cambiamento dello stato dell'automa secondo la sua funzione di transizione.

Figura 2: Implementazione Java del DFA di Figura 1.

# Esercizio 1.1 - Implementazione

```
public class TreZeri
{
    public static boolean scan(String s)
    {
        int state = 0;
        int i = 0;

        while (state >= 0 && i < s.length()) {
            final char ch = s.charAt(i++);

            switch (state) {
                case 0:
                    if (ch == '0')
                        state = 1;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 1:
                    if (ch == '0')
                        state = 2;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 2:
                    if (ch == '0')
                        state = 3;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 3:
                    if (ch == '0' || ch == '1')
                        state = 3;
                    else
                        state = -1;
                    break;
            }
        }

        return state == 3;
    }

    public static void main(String[] args)
    {
        System.out.println(scan(args[0]) ? "OK" : "NOPE");
    }
}
```

Osserva: viene assegnato il valore **-1** alla variabile state se viene incontrato un simbolo diverso da 0 e 1.

Tale valore non è uno stato valido, ma rappresenta una condizione di errore irrecuperabile.

Figura 2: Implementazione Java del DFA di Figura 1.

# Esercizio 1.1 - Implementazione

```
public class TreZeri
{
    public static boolean scan(String s)
    {
        int state = 0;
        int i = 0;

        while (state >= 0 && i < s.length()) {
            final char ch = s.charAt(i++);

            switch (state) {
                case 0:
                    if (ch == '0')
                        state = 1;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 1:
                    if (ch == '0')
                        state = 2;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 2:
                    if (ch == '0')
                        state = 3;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

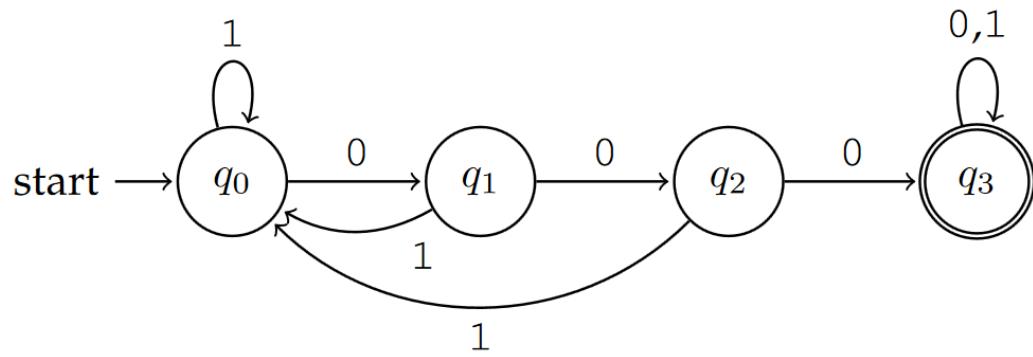
                case 3:
                    if (ch == '0' || ch == '1')
                        state = 3;
                    else
                        state = -1;
                    break;
            }
        }

        return state == 3;
    }

    public static void main(String[] args)
    {
        System.out.println(scan(args[0]) ? "OK" : "NOPE");
    }
}
```

Figura 2: Implementazione Java del DFA di Figura 1.

- Nel caso in cui **state = -1**, il metodo scan restituisce subito il valore **false** (non legge altri simboli dal input).
- Osserva: il metodo scan restituisce:
  - **true** se l'input (1) è costituito solo da simboli che appartiene all'alfabeto, e (2) è accettato dal DFA.
  - **false** se l'input (1) è costituito solo da simboli che appartiene all'alfabeto, e (2) non è accettato dal DFA.
  - **false** se l'input contiene almeno un simbolo che non appartiene all'alfabeto.



```

public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

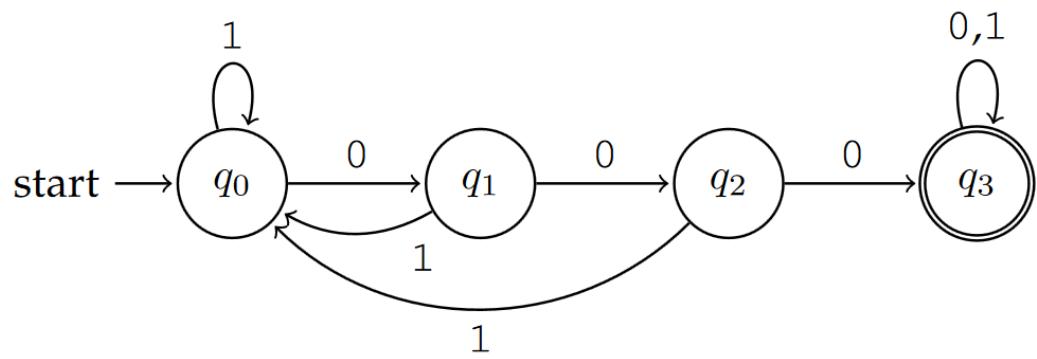
        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
  
```

Stringa: 0 1 0 0 0 1



```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

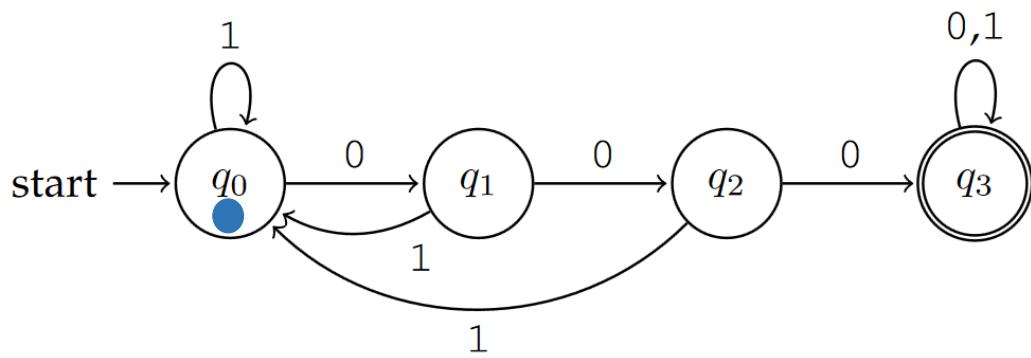
        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }
    return state == 3;
}
```

Stringa: 0 1 0 0 0 1



```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

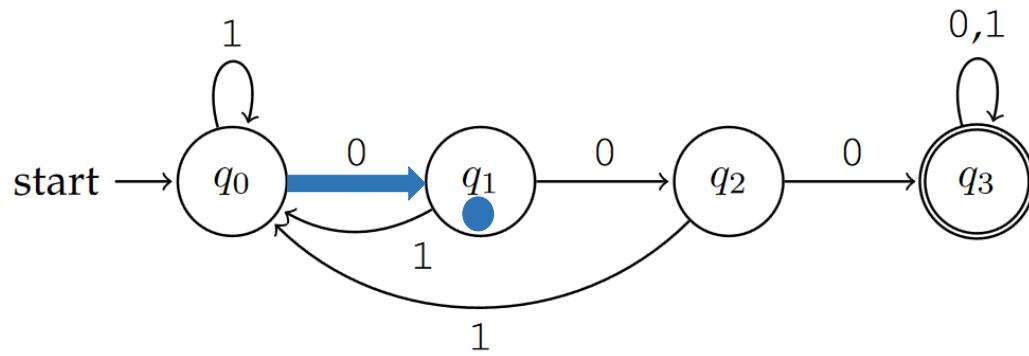
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
```

state = 0  
i = 0

Stringa: 0 1 0 0 0 1



```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

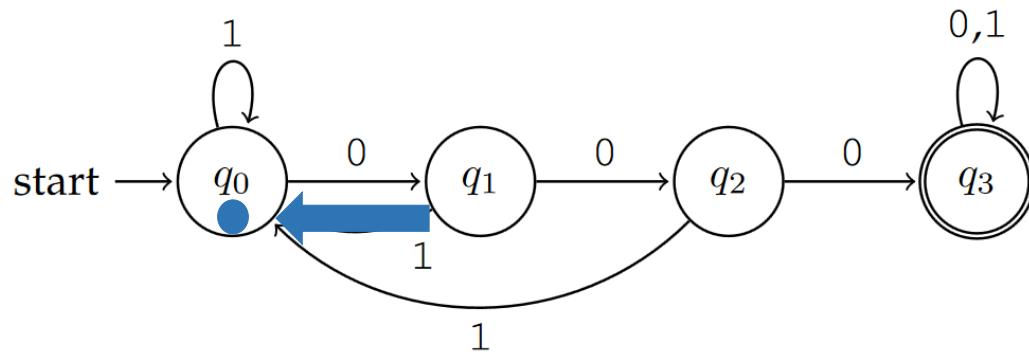
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
```

state = 1  
i = 1

Stringa: 0 **1** 0 0 0 1



```

public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

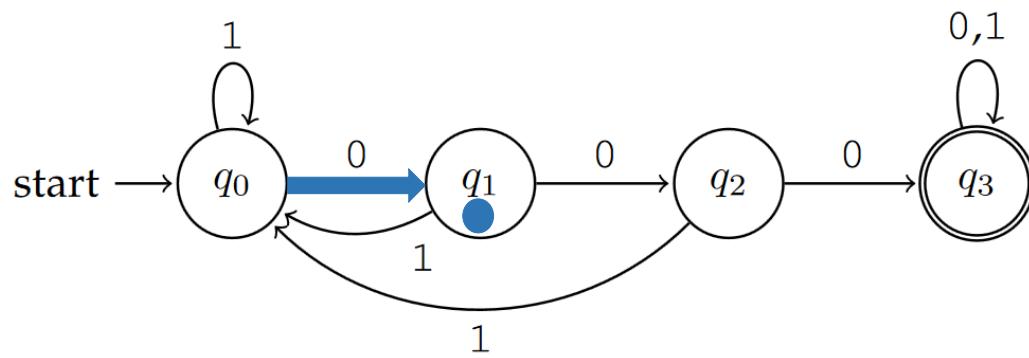
        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}

```

state = 0  
i = 2

Stringa: 0 1 0 0 0 1



```

public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

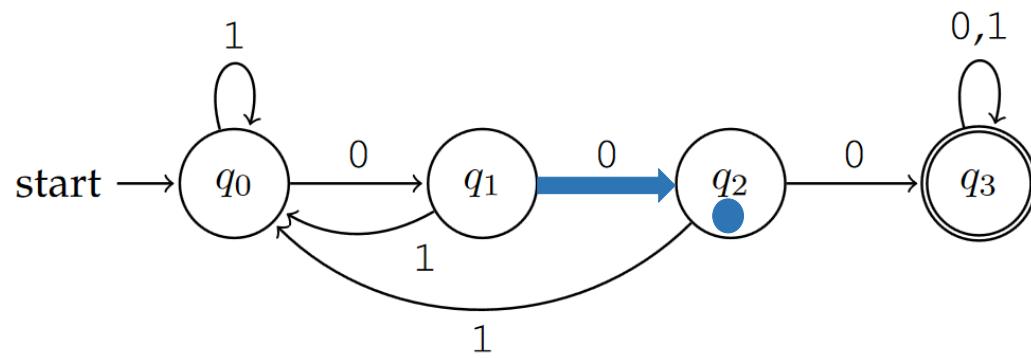
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
  
```

state = 1  
i = 3

Stringa: 0 1 0 0 0 1



```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

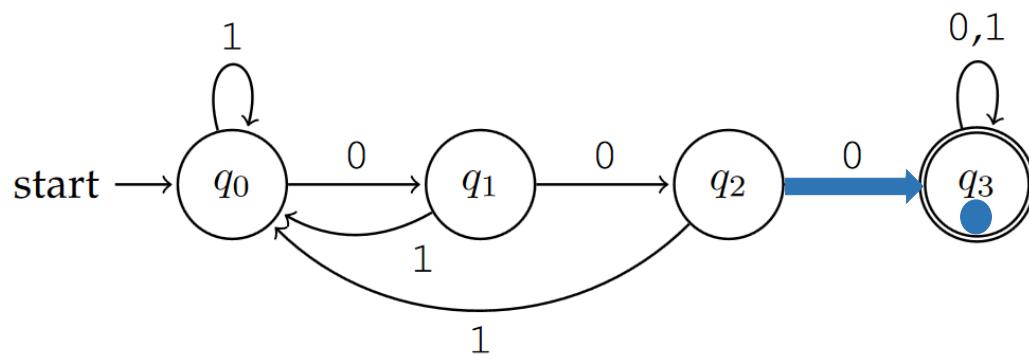
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
```

state = 2  
i = 4

Stringa: 0 1 0 0 0 1



```

public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

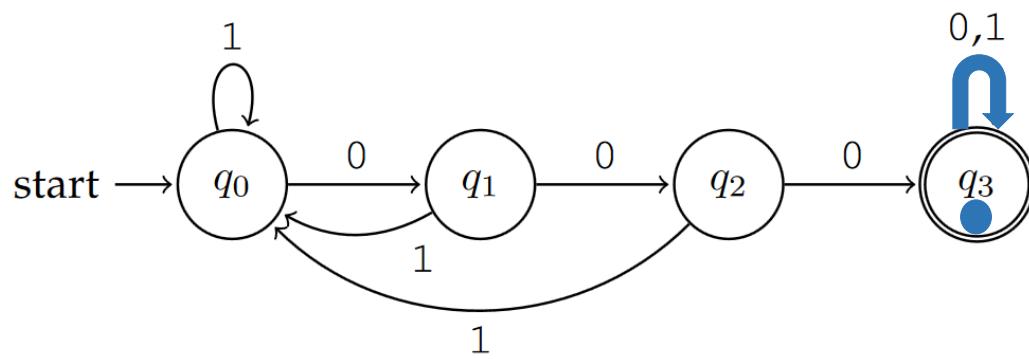
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
  
```

state = 3  
i = 5

Stringa: 0 1 0 0 0 1



```

public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

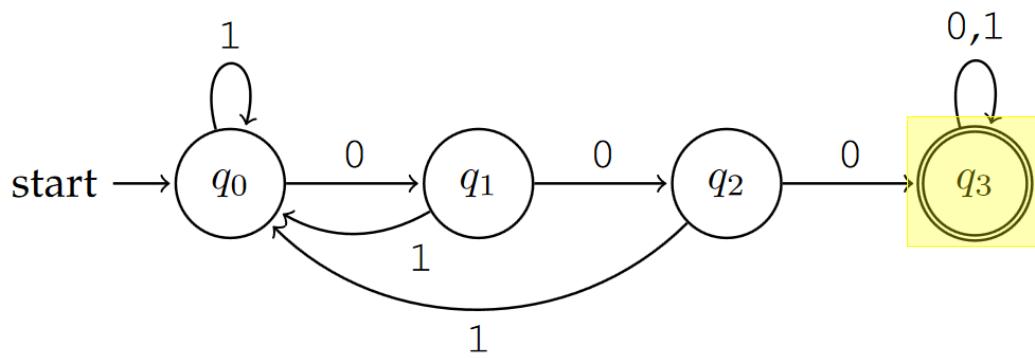
            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
  
```

state = 3  
i = 6

Stringa: 0 1 0 0 0 1



```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
            case 0:
                if (ch == '0')
                    state = 1;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 1:
                if (ch == '0')
                    state = 2;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 2:
                if (ch == '0')
                    state = 3;
                else if (ch == '1')
                    state = 0;
                else
                    state = -1;
                break;

            case 3:
                if (ch == '0' || ch == '1')
                    state = 3;
                else
                    state = -1;
                break;
        }
    }

    return state == 3;
}
```

# Esercizio 1.1 - Implementazione

```
public class TreZeri
{
    public static boolean scan(String s)
    {
        int state = 0;
        int i = 0;

        while (state >= 0 && i < s.length()) {
            final char ch = s.charAt(i++);

            switch (state) {
                case 0:
                    if (ch == '0')
                        state = 1;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 1:
                    if (ch == '0')
                        state = 2;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 2:
                    if (ch == '0')
                        state = 3;
                    else if (ch == '1')
                        state = 0;
                    else
                        state = -1;
                    break;

                case 3:
                    if (ch == '0' || ch == '1')
                        state = 3;
                    else
                        state = -1;
                    break;
            }
        }

        return state == 3;
    }

    public static void main(String[] args)
    {
        System.out.println(scan(args[0]) ? "OK" : "NOPE");
    }
}
```

Output del programma:

- stampa OK sul terminale se la stringa di input è accettato da  $A$
- NOPE altrimenti

Input:

dalla linea di comando,  
come argomento (**tra virgolette**):

```
C:\Users\sproston\Documenti\LFT1920\CommandLine>java TreZeri "010"
NOPE

C:\Users\sproston\Documenti\LFT1920\CommandLine>java TreZeri "010001"
OK
```

Figura 2: Implementazione Java del DFA di Figura 1.

# Esercizio 1.1

## Implementazione e Test

### Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

- Scaricare il codice TreZeri.java dalla pagina del corso (fig. 1 del documento)
- Compilarlo e **testarlo** su un insieme significativo di stringhe:
  - “010101”, “1100011001”, “10214”, “10002100”, etc.

# Esercizio 1.1 bis

Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

- Come deve essere modificato il DFA in Figura 1 per riconoscere il linguaggio complementare, ovvero il linguaggio delle stringhe di 0 e 1 che non contengono 3 zeri consecutivi?
- Progettare e implementare il DFA modificato, e testare il suo funzionamento

# Esercizio 1.2

- Progettare e implementare un DFA che riconosca il linguaggio degli identificatori in un linguaggio in stile Java
- Un identificatore è una sequenza non vuota di lettere, numeri, ed il simbolo di “underscore” \_ che:
  - non comincia con un numero
  - non può essere composto solo dal simbolo \_
- Compilare e testare il suo funzionamento su un insieme significativo di esempi.

# Esercizio 1.2

## Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

- Ad esempio, il DFA

- deve accettare le stringhe:  
“x”, “flag1”, “x2y2”, “x\u03041”, “lft\_lab”,  
“temp\u0304”, “x\u03041\u0304y\u03042\u0304”, “x\u0304\u0304”,  
“\_\u0304\_\u0304 Pippo”, “A\u03041”

YES  
 NO

- ma non deve accettare: “5”,  
“221B”, “123”, “9\u0304 to \_5”,  
“\_\u0304\_\u0304”, “\_\u0304”, “\_\u0304\_\u0304”

YES  
 NO

# Esercizio 1.3

- Progettare e implementare un DFA che riconosca il linguaggio di stringhe che contengono un **numero di matricola** seguito (**subito**) da un **cognome**, dove la combinazione di matricola e cognome corrisponde a studenti del turno **2** o del turno **3** del laboratorio di LFT.
- Regole per suddivisione di studenti in turni:
  - Turno 1: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è dispari;
  - Turno 2: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è pari;
  - Turno 3: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è dispari;
  - Turno 4: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è pari.
- Esempio:
  - “123456Bianchi” e “6599321Rossi” sono stringhe del linguaggio,
  - “654321Bianchi” e “1277456Rossi” non sono stringhe del linguaggio.
- Nel contesto di questo esercizio, un **numero di matricola** non ha un numero prestabilito di cifre ma deve essere composto di **almeno una cifra**.
- Un **cognome** comincia con una lettera maiuscola, seguita (eventualmente) da lettere minuscole e deve essere composto di **almeno una lettera maiuscola**.
- L'automa deve accettare le stringhe:
  - “2Bianchi” e “122B”  
ma non “654321” e “Rossi”
  - YES
  - NO
  - YES
  - NO
- *Assicurarsi che il DFA sia minimo (non ora)*

Su alcuni esercizi ha senso richiedere  
che i DFA siano minimi:  
ripenseremo a questo aspetto  
in seguito

# Esercizio 1.3

Metodo:

- 1) Progettazione dell'automa
- 2) Implementazione in Java
- 3) Test su esempi di input

# Esercizio 1.3 bis

- Progettare e implementare un DFA che riconosca il linguaggio di stringhe che contengono un **numero di matricola** seguito (**subito**) da un **cognome**, dove la combinazione di matricola e cognome corrisponde a studenti del turno **1** o del turno **4** del laboratorio di LFT.
- Regole per suddivisione di studenti in turni:
  - Turno 1: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è dispari;
  - Turno 2: cognomi la cui iniziale è compresa tra A e K, e il numero di matricola è pari;
  - Turno 3: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è dispari;
  - Turno 4: cognomi la cui iniziale è compresa tra L e Z, e il numero di matricola è pari.
- Esempio:
  - “123456Bianchi” e “654321Rossi” non sono stringhe del linguaggio,
  - “654321Bianchi” e “123456Rossi” sono stringhe del linguaggio.
- Nel contesto di questo esercizio, un **numero di matricola** non ha un numero prestabilito di cifre ma deve essere composto di **almeno una cifra**.
- *Assicurarsi che il DFA sia minimo (non ora).*

# Esercizio 1.4

- Modificare l'automa dell'esercizio precedente in modo che riconosca le combinazioni di matricola e cognome di studenti del turno 2 o del turno 3 del laboratorio, dove il numero di matricola e il cognome:
    - possono essere separati da una sequenza di spazi (simbolo ws - spazio)
    - possono essere precedute e/o seguite da sequenze eventualmente vuote di spazi.
  - Per esempio, l'automa
    - deve accettare la stringa “654321 Rossi” e “123456 Bianchi” (dove, nel secondo esempio, ci sono spazi prima del primo carattere e dopo l'ultimo carattere)
    - ma non “1234 56Bianchi” e “123456Bia nchi”.
- 1) Provate prima a pensare a una soluzione senza considerare cognomi composti
- 2) Poi progettare una variante dell'automa in cui i cognomi composti (con un numero arbitrario di parti) possono essere accettati:
- Per esempio, la stringa “123456De Gasperi” deve essere accettata.
  - Modificare l'implementazione Java dell'automa di conseguenza.
- 3) (facoltativo - variante semplice) Modificate l'automa dell'esercizio in modo che riconosca le combinazioni di matricola e cognome di studenti del turno 1 o del turno 4 del laboratorio (per il resto, per quanto riguarda il white space stessi vincoli sull'accettazione delle stringhe)

# Esercizio 1.5

- Esercizio 1.5. Progettare e implementare un DFA che, come in Esercizio 1.3, riconosca il linguaggio di stringhe che contengono **matricola** e **cognome** di studenti del **turno 2 o del turno 3** del laboratorio, ma in cui il **cognome precede il numero di matricola**
  - in altre parole, le posizioni del cognome e matricola sono scambiate rispetto all’Esercizio 1.3).
- *Assicurarsi che il DFA sia minimo (non ora).*