Chapter 1

Exercise 1

back-end	maps code into computer specific code
front-end	understands code syntax and checks for errors
grammar	rules of the language
instruction scheduling	choosing the order of the instructions
instruction selection	choosing which instructions to use
optimiser	a transformer that improves the IR
parsing	grouping of tokens based on grammar
register allocation	optimises the registers used in the program
scanning	transforms the code language into tokens
type checking	checking if the groups of tokens are meaningful

Exercise 2

Most	students	is	good	programmers	.
adj	noun	verb	adj	noun	end
Modifier	noun	verb	**Modifier**	noun	end
Subject		verb	**Object**		end
Sentence					

2. *is* should be *are* because 'students' is plural. This could be compared to a type error.

3. Parsing and type checking

Exercise 3

Sentence	→	**Subject** verb **Object** endmark
Subject	→	noun
Subject	→	particle noun
Subject	→	**Modifier** noun
Subject	→	particle **Modifier** noun
Object	→	noun
Object	→	particle noun
Object	→	**Modifier** noun
Object	→	particle **Modifier** noun
Modifier	→	adjective
Modifier	→	**Modifier** adjective

Exercise 4

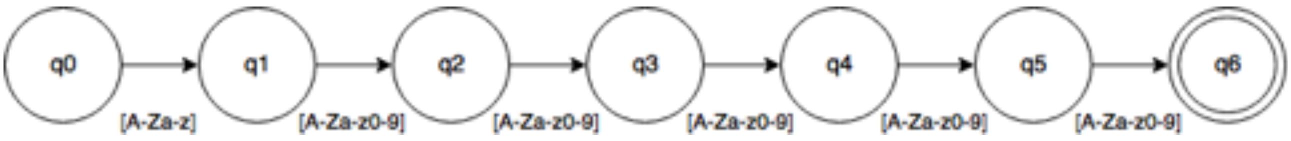
```
1. d ← d + 2 * (a + b)
1 3   loadAI     r(arp), @a => r(a)
2 4   loadAI     r(arp), @b => r(b)
3 5   loadAI     r(arp), @d => r(d)
5 5   add        r(a), r(b)  => r(a)
6 6   add        r(a), r(a)  => r(a)
7 7   add        r(d), r(a)  => r(d)
8 10  storeAI    r(d)  => r(arp), @d
```

```
2.
1   3  loadAI      r(arp), @a => r(1)
2   4  loadAI      r(arp), @b => r(2)
5   5  add         r(1), r(2) => r(1)
6   6  add         r(1), r(1) => r(1)
7   9  loadAI      r(arp), @d => r(2)
10  10 add         r(1), r(2) => r(1)
11  13 storeAI     r(1)  => r(arp), @d
```

```
3.
1 3   loadAI      r(arp), @a => r(1)
2 4   loadAI      r(arp), @b => r(2)
3 5   loadAI      r(arp), @d => r(3)
5 5   add         r(1), r(2) => r(1)
6 6   add         r(1), r(1) => r(1)
7 7   add         r(d), r(a) => r(d)
8 10  storeAI     r(d)  => r(arp), @d
```

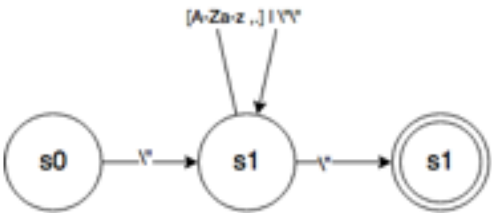
Chapter 2

Exercise 5



Exercise 6

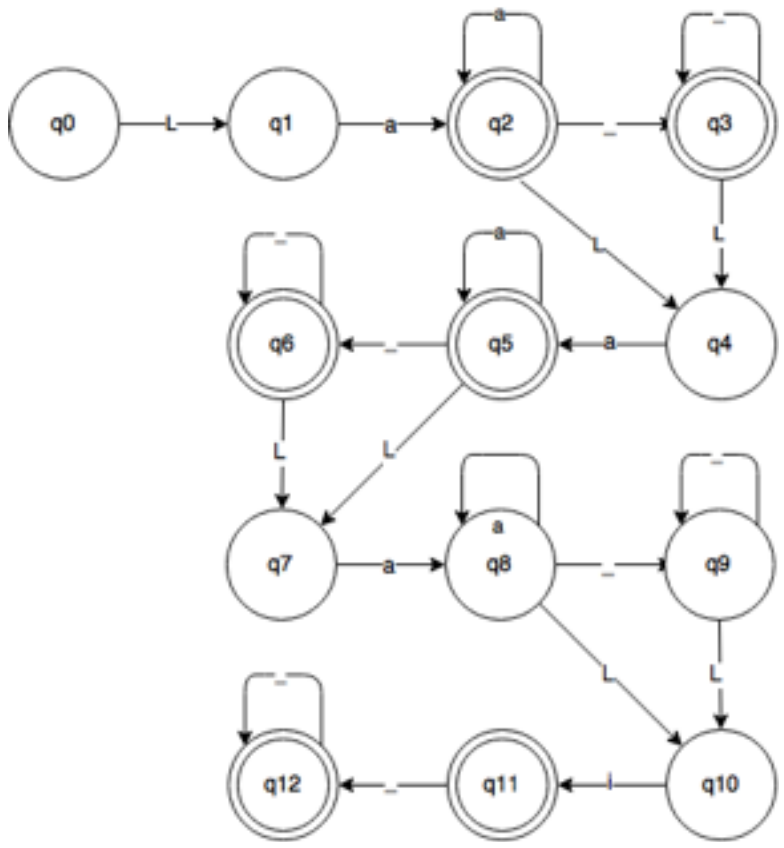
```
regex = \"([A-Za-z ,.]|\"{2})*\"
```



Exercise 7

1. (La+\ *)
(La+\ *){2}
(La+\ *){3}(Li\ *)

2.



3.

- ‘Laaaa La‘ + ‘Laa’
- ‘La ‘ + ‘La La La Li’