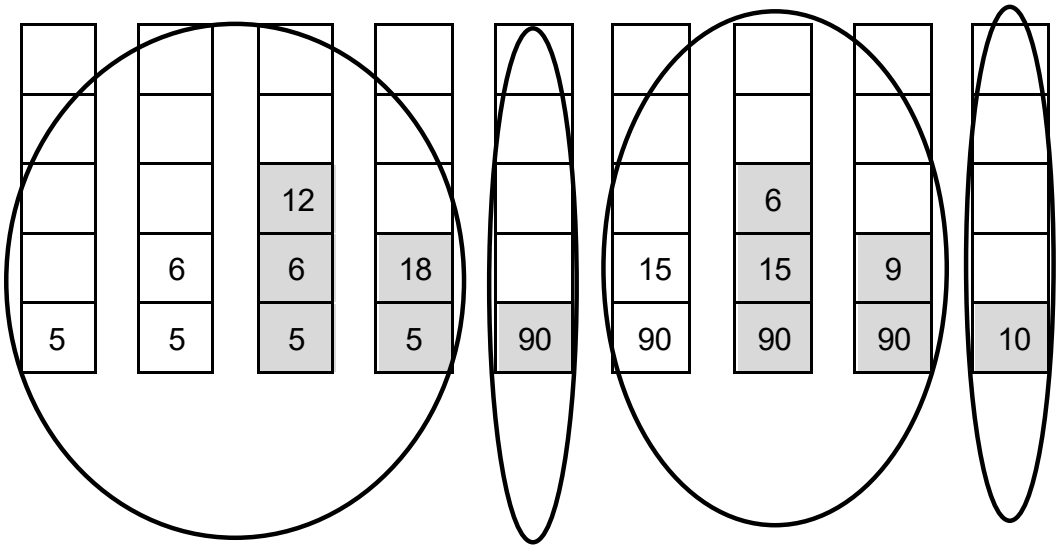
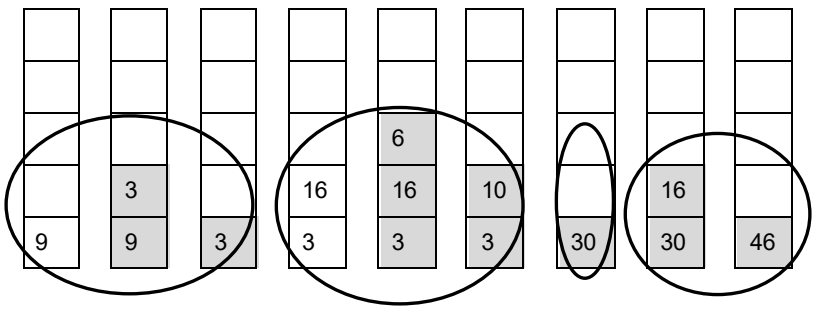
8(a)	**One** mark per mark point (**Max 2**) MP1 To convert the high-level source code / program into a sequence of tokens MP2 … that can be sent to the parser for syntax analysis MP3 To create a symbol table MP4 To remove the unnecessary white space and comments from the code
8(b)	**One** mark `2 6 -` **One** mark `13 7 + * 5 /` Complete answer `2 6 - 13 7 + * 5 /`
8(c)	**One** mark per ring (**Max 4**) 

8(a)	To improve the code by making it use minimum resources (CPU, memory/storage, time) // Answer by example: To improve the code by • minimising program storage • minimising CPU time • minimising memory use • minimising program execution time (includes peripheral use as well as CPU)
8(b)	**One** mark `(a - b + c)` **One** mark `* (c - a)` **One** mark `/ d` Complete answer `(a - b + c) * (c - a) / d`
8(c)	**One** mark per ring (**Max 4**) 

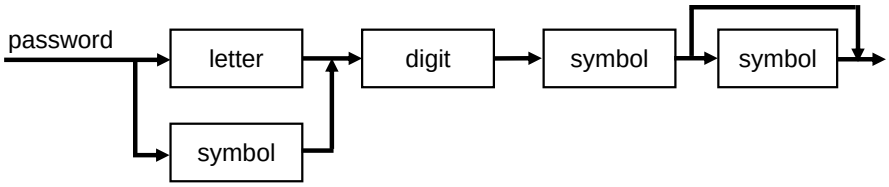
6	**One** mark for each mark point (**Max 4**) MP1 The interpreter translates the source code one line at a time MP2 If the line is syntax error free it is executed MP3 It is not stored in executable format MP4 If an error is found, the program halts with an error message MP5 Each line must be translated every time it is run, including lines running multiple times for example in loops

7(a)	**One** mark for each correct answer **#Jd7** – must begin with a member of the group uppercase // cannot begin with a symbol **C%6A** – the fourth character cannot be a member of the group uppercase // the fourth character must be either a symbol, digit or lowercase

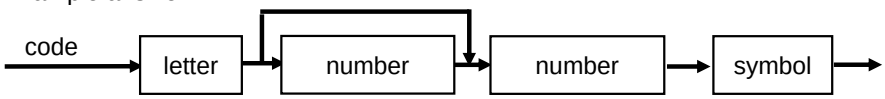
7(b)	**One** mark **per** mark point
	`<uppercase> ::= A | C | E | G | J`
	`<passcode> ::= <uppercase><code>`
	`<code> ::= <lowercase>|<symbol>|<digit>`
	`|<lowercase><code>|<symbol><code>|<digit><code>`

8	**One** mark per mark point (**Max 4**) MP1 It is the first stage of compilation MP2 White space and comments are removed MP3 It takes modified source code and breaks it into a series of tokens MP4 Each token is categorised and assigned types MP5 Identifiers are stored in a symbol table MP6 If the lexical analyser finds invalid tokens, it generates an error MP7 Acceptable data from the lexical analyser passes to the syntax analyser

8	**One** mark per mark point (**Max 4**) MP1 It is the second stage of compilation // It's the compilation stage after lexical analysis. MP2 It takes input from the lexical analyser in the form of token streams. MP3 The source code is analysed / parsed against the rules of the language to detect any errors in the code. MP4 The output from this phase is a parse tree. MP5 Syntax errors are reported.

10(a)	**One** mark per mark point (**Max 4**) • `<operator> ::= + | - | * | /` • `<label> ::= <letter><digit>|<letter><digit><digit>` • `<equation> ::= <label> =` • `<label><operator><label>`
10(b)(i)	**One** mark per mark point (**Max 3)** MP1 begin with either a letter or a symbol MP2 end with either one or two symbols MP3 digit and all other connections and label correct. 

10(b)(ii)	**One** mark per mark point (**Max 2)**
	
	• `<password> ::= <letter><digit><symbol>|`
	• `<letter><digit><symbol><symbol>|<symbol><digit><symbol>|`
	`<symbol><digit><symbol><symbol>`
	
	`<password> ::= <letter><digit><symbol>|`
	`<letter><digit><symbol><symbol>|<symbol><digit><symbol>|`
	`<symbol><digit><symbol><symbol>`
	
	Alternative Answer
	
	One mark per mark point (**Max 2)**
	• All three lines correct
	• Any two lines correct
	
	`<first> ::= <letter>|<symbol>`
	`<last> ::= <symbol>|<symbol><symbol>`
	`<password> ::= <first><digit><last>`

7(a)	**One** mark per mark point (**Max 2)** MP1 21 - a number must begin with an odd digit, 2 is even MP2 123 - a number can only be one or two digits in length not three
7(b)	**One** mark per mark point (**Max 2)** MP1 `<symbol> ::= % | £ | # | @ | $` MP2 `<number> ::= <odd>|<odd><even>|<odd><odd>`
7(c)(i)	**One** mark per mark point (**Max 3)** MP1 letter, number and symbol all included in correct order: letter first, followed by number, finishing with symbol MP2 provision for one or two numbers including relevant connectors MP3 all other connections and label correct and no additional data Example answer 

7(c)(ii)	**One** mark per correct line (**Max 2)**
	
	`<code> ::= <letter><number><symbol>|`
	`<letter><number><number><symbol>`
	
	OR
	
	`<code> ::= <letter><number><number><symbol>|`
	`<letter><number><symbol>`
	
	Alternative Answer
	
	One mark per correct line (**Max 2)**
	
	`<digits> ::= <number>|<number><number>`
	`<code> ::= <letter><digits><symbol>`