Past-paper walk-through

Translation Software ▪ 16.2

eXercise

Questions in this set

- 9618/31 N25 Q8 ▪ 8 marks
- 9618/32 N25 Q8 ▪ 8 marks
- 9618/31 J25 Q6 ▪ 4 marks
- 9618/31 J25 Q7 ▪ 6 marks
- 9618/32 J25 Q8 ▪ 4 marks
- 9618/33 J25 Q8 ▪ 4 marks
- 9618/31 N24 Q10 ▪ 9 marks
- 9618/32 N24 Q7 ▪ 9 marks

Reading the mark scheme

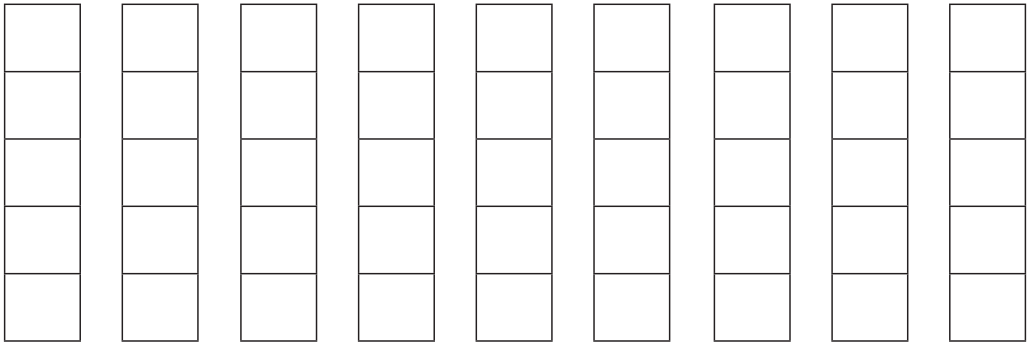
- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 8

9618/31 N25 ▪ Q8 ▪ 8 marks

(a) Outline the purpose of lexical analysis during the compilation of a program. **[2]**

Question 8



Question 8

9618/31 N25 ▪ Q8 ▪ 8 marks

(b) Write the Reverse Polish Notation (RPN) for the given infix expression:
$(2 - 6) * (13 + 7)/5$ **[2]**

Question 8

9618/31 N25 ▪ Q8 ▪ 8 marks

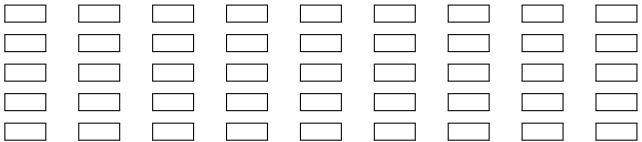
(c) The RPN expression:

```
d a b + * c a - /
```

is to be evaluated, where:

$a = 6$, $b = 12$, $c = 15$ and $d = 5$.

Show the changing contents of the stack as the RPN expression is evaluated.



Question 8

[4]

Solution 8

(a) Mark scheme

- 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

Solution 8

- To remove the unnecessary white space and comments from the code

Solution 8

(b)

Mark scheme

- One mark 2 6 –
- One mark 13 7 + * 5 /
- Complete answer
- 2 6 – 13 7 + * 5 /

Solution 8

(c) Mark scheme

- One mark per ring (Max 4)
- 12
- 6
- 6
- 6
- 18
- 15
- 15

Solution 8

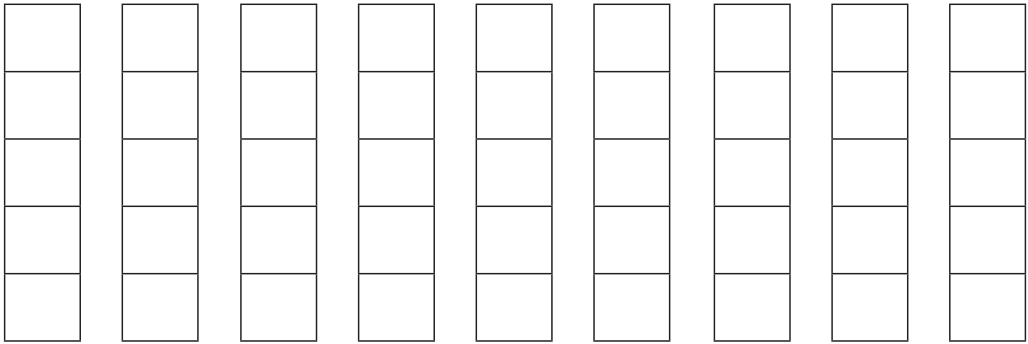
- 9
- 5
- 5
- 5
- 5
- 90
- 90
- 90
- 90
- 10

Question 8

9618/32 N25 ▪ Q8 ▪ 8 marks

(a) State the purpose of the optimisation stage in the compilation of a program. **[1]**

Question 8



Question 8

9618/32 N25 ▪ Q8 ▪ 8 marks

(b) Convert this Reverse Polish Notation (RPN) back to its original infix form:

`a b - c + c a - * d /` **[3]**

Question 8

9618/32 N25 ▪ Q8 ▪ 8 marks

(c) The RPN expression:

`c a / b d - * b +`

is to be evaluated, where:

$a = 3$, $b = 16$, $c = 9$ and $d = 6$.

Show the changing contents of the stack as the RPN expression is evaluated.

Question 8

Question 8

Question 8

Question 8

Question 8

Question 8

Question 8

Question 8

Question 8

Question 8

[4]

Solution 8

(a)

Mark scheme

- 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)

Solution 8

(b) Worked steps

Work from **left to right with a stack**. Push every operand; when an operator appears, pop the **last two** operands, combine them in that order, and push the bracketed result back.
For `a b - c + c a - * d /`:

Token	Stack after
a, b	a, b
-	(a - b)
c	(a - b), c
+	(a - b + c)
c, a	(a - b + c), c, a
-	(a - b + c), (c - a)
*	(a - b + c) * (c - a)
d	(a - b + c) * (c - a), d
/	(a - b + c) * (c - a) / d

Solution 8

So the infix form is $(a - b + c) \times (c - a) \div d$.
Three marks: `(a - b + c)`, `* (c - a)`, and `/ d`.
Two things decide whether the answer is right:

- **Order matters for - and /.** The **second** operand popped is the left-hand side: popping `a` then `c` from … `c a -` gives `c - a`, not `a - c`. Getting this backwards is the single commonest error, and it is invisible unless you check.
- **Bracket each result as you push it.** The brackets are what preserve the order of evaluation when the sub-expression is later combined; dropping them turns `(a - b + c) * (c - a)` into `a - b + c * c - a`, which is a different expression entirely.

Solution 8

RPN needs no brackets precisely because the order is fixed by position — so converting back is the job of putting them in. **Mark scheme**

- One mark (a – b + c)
- One mark * (c - a)
- One mark / d
- Complete answer
- (a – b + c) * (c - a) / d
- 3
- October/November
- 2025

Solution 8

(c) Mark scheme

- One mark per ring (Max 4)
- 6
- 3
- 16
- 16
- 10
- 16
- 9

Solution 8

- 9
- 3
- 3
- 3
- 3
- 30
- 30
- 46

Question 6

9618/31 J25 ▪ Q6 ▪ 4 marks

Describe the process of executing a program using an interpreter. **[4]**

Solution 6

Mark scheme

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

Question 7

9618/31 J25 ▪ Q7 ▪ 6 marks

Several syntax diagrams are shown.

uppercase

- A
- C
- E
- G
- J

lowercase

- b
- d

Question 7

- f
- h
- k

symbol

- $
- @
- #
- &
- %

digit

- 0

Question 7

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

passcode

- uppercase

Question 7

- lowercase
- symbol
- digit

(a) State why each **passcode** is invalid for the given syntax diagrams.

`#Jd7`

`C%6A` **[2]**

(b) Complete the Backus-Naur Form (BNF) for `<uppercase>` and `<passcode>`. **[4]**

Solution 7

(a)

Mark scheme

- **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

Solution 7

(b) Mark scheme

- **One mark per mark point**
-
- <uppercase> ::= A | C | E | G | J
- <passcode> ::= <uppercase><code>
- <code> ::= <lowercase>|<symbol>|<digit>
- |<lowercase><code>|<symbol><code>|<digit><code>
-

Question 8

9618/32 J25 ▪ Q8 ▪ 4 marks

Explain what is meant by **lexical analysis** during program compilation. **[4]**

Solution 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

Solution 8

- 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

Question 8

9618/33 J25 ▪ Q8 ▪ 4 marks

Explain the process of **syntax analysis** during program compilation. **[4]**

Solution 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

Solution 8

- 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.

Question 10

9618/31 N24 ▪ Q10 ▪ 9 marks

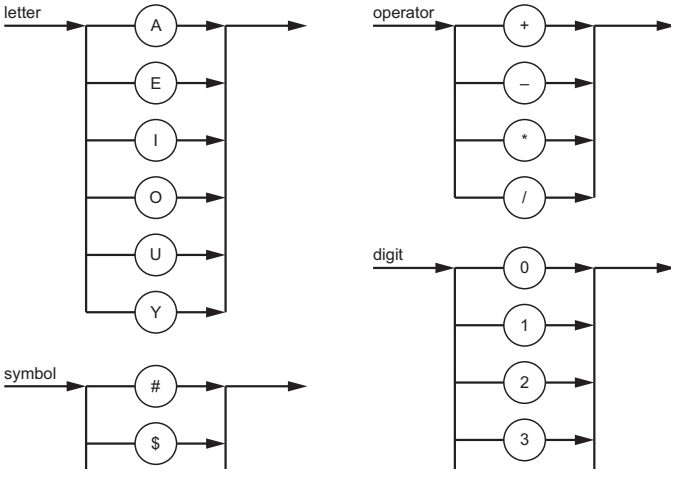
Several syntax diagrams are shown.

(a) Complete the Backus-Naur Form (BNF) for the given syntax diagrams. **[4]**

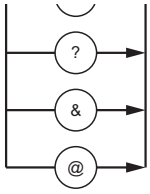
(b)(i) Draw a syntax diagram for **password**. **[3]**

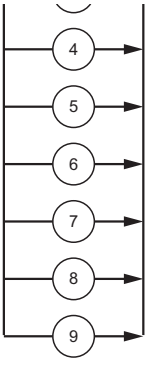
(b)(ii) Write the BNF for **password**. **[2]**

Question 10



Question 10





Question 10

Solution 10

(a)

Mark scheme

- One mark per mark point (Max 4)
- ▪ <operator> ::= + | – | * | /
- ▪ <label> ::= <letter><digit>|<letter><digit><digit>
- ▪ <equation> ::= <label> =
- ▪ <label><operator><label>

Solution 10

(b)(i) Worked steps

Read the rule and turn each phrase into a piece of the diagram, then connect them left to right. Three marks:

1. It must **begin with either a letter or a symbol** — two parallel routes into the diagram, converging before what follows.
2. It must **end with one or two symbols** — a `symbol` box with a loop back around it, so the reader may pass through once or twice.
3. **The `digit` box and all remaining connections and the label are correct.**

Solution 10

The two notations that earn the marks:

- **Alternatives** are parallel paths that split and rejoin. Both must return to the same point, or the diagram allows something the rule does not.
- **Repetition** is a path that loops back. A loop with no limit permits any number of repeats, so "one or two" needs either a bounded arrangement — two boxes in sequence with a bypass around the second — or an explicit note.

Label the whole diagram `password` at its entry. And add nothing else: a mark point says the remaining connections must be correct with no additional data, so an extra path that allows a password of digits only would cost it even though everything required is drawn.

Solution 10

Trace the finished diagram with two examples before you leave it — one that should be accepted and one that should not. Any route through the diagram is a legal password, so if the invalid example finds a path, the diagram is wrong.

Mark scheme

- 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.
- 3 symbol symbol symbol password letter digit

Solution 10

(b)(ii) Mark scheme

- 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

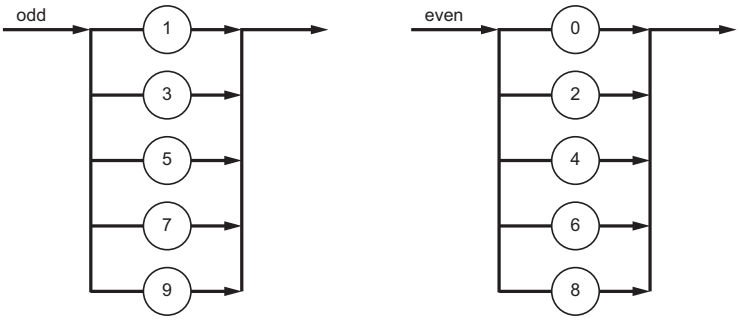
Solution 10

- 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>

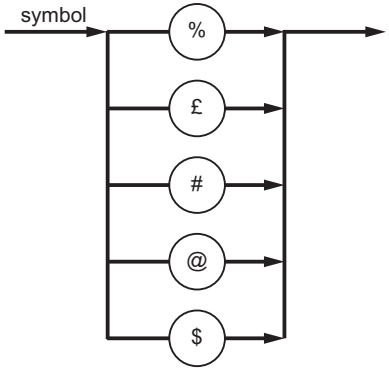
Question 7

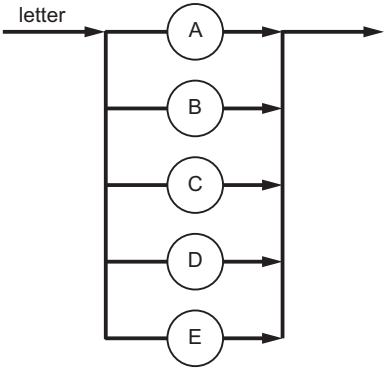
9618/32 N24 ▪ Q7 ▪ 9 marks

7 Several syntax diagrams are shown.

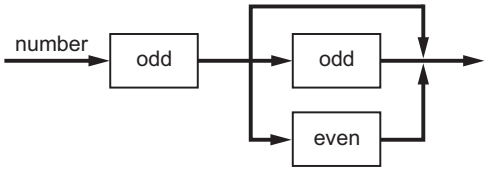


Question 7





Question 7



(a) State why each number is invalid for the given syntax diagrams.

```
21
```

Question 7

```
123
```

Question 7

[2]

(b) Complete the Backus-Naur Form (BNF) for the given syntax diagrams.

Question 7

[2]

(c) A new syntax rule, **code**, is required. It must begin with a letter, followed by one or two numbers, and end with a symbol.

(i) Draw a syntax diagram for **code**.

Question 7

[3]

(ii) Write the BNF for **code**.

Solution 7

(a)

Mark scheme

- 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
- 2

Solution 7

(b) Worked steps

BNF and a syntax diagram say the same thing in different notations, so translate structure for structure:

- **A choice between alternatives** in the diagram (parallel branches) becomes | in BNF.
- **A sequence** (one box after another) becomes items written side by side.
- **A repetition** (a loop back) becomes a recursive rule.

Two marks:

```
<symbol> ::= % | £ | # | @ | $
<number> ::= <odd> | <odd><even> | <odd><odd>
```

Solution 7

Every alternative must appear. `<number>` allows a single odd digit, or an odd followed by an even, or an odd followed by another odd — three branches, so three alternatives separated by |.

Two conventions carry marks in this topic: **angle brackets** around the name of every rule being referred to, and ::= (not =) as the defining symbol. A rule that refers to odd without brackets is referring to a literal three-letter string.

Mark scheme

- One mark per mark point (Max 2)
- MP1 <symbol> ::= % | £ | # | @ | $
- MP2 <number> ::= <odd>|<odd><even>|<odd><odd>
- 2

Solution 7

(c)(i) Worked steps

Draw the diagram from the BNF the same way, in reverse. Three marks:

1. **`letter`, `number` and `symbol` in the correct order** — letter first, then number, finishing with symbol.
2. **Provision for one or two numbers**, with the connectors that allow both.
3. **All other connections and the label correct**, with no extra data.

Solution 7

"One or two" is drawn as a **loop back** around the `number` box — a path that lets the reader go round once more — or as two parallel routes, one through a single `number` and one through two in sequence. Either is accepted; an unbounded loop is not, because it would permit three.

The mark for "no additional data" is worth reading twice. An extra box, or an extra path that allows something the rule does not, loses it even when everything required is present.

Label the whole diagram with the rule it defines — here `code` — at the entry point. An unlabelled diagram defines nothing. **Mark scheme**

- One mark per mark point (Max 3)

Solution 7

- 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
- 3
- letter
- number
- symbol
- number

Solution 7

- code
- 9618/32
- October/November 2024

Solution 7

(c)(ii) Worked steps

Write the BNF rule the diagram from part (c)(i) defines. Two marks, one per correct line.

```
<code> ::= <letter><number><symbol> | <letter><number><number><symbol>
```

Solution 7

An equally acceptable answer factors the repetition into a rule of its own, which is neater:

```
<digits> ::= <number> | <number><number>
<code>   ::= <letter><digits><symbol>
```

The translation rules are always the same three:

- **Sequence** — items written side by side.
- **Choice** — alternatives separated by |.
- **Repetition** — either write the alternatives out (as the first answer does) or introduce a helper rule (as the second does). Cambridge pseudocode BNF has no * or + operator, so "one or two" must be spelled out one way or the other.

Solution 7

Two conventions carry the marks: **angle brackets** around every rule name being referred to, and `::=` as the defining symbol. `code = letter number symbol` refers to three literal strings and defines nothing.

Solution 7

Check the finished rule against the diagram by generating one example from each alternative. If either produces something the diagram would reject, they do not describe the same language. **Mark scheme**

- 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>