Pseudocode and flowcharts

Algorithms and programming (Paper 2) · revision notes

Algorithms can be written as pseudocode, as flowcharts, in OCR Exam Reference Language or in a high-level language. You need to create, read, complete, correct and refine algorithms in these forms. Expect flowchart symbols, completing code with gaps and writing short algorithms.

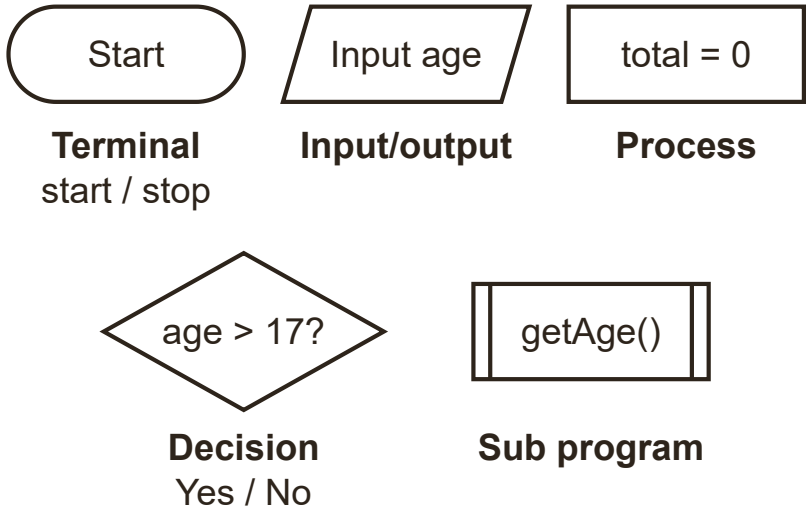
Key facts

- Rounded rectangle = terminal (Start / Stop)
- Rectangle = process (calculation or assignment)
- Rectangle with double side lines = sub program
- Pseudocode: English-like steps, one per line, indented
- Parallelogram = input or output
- Diamond = decision (two exits: Yes / No)
- Arrow back to an earlier step = a loop
- ERL = OCR Exam Reference Language, used for code in the exam

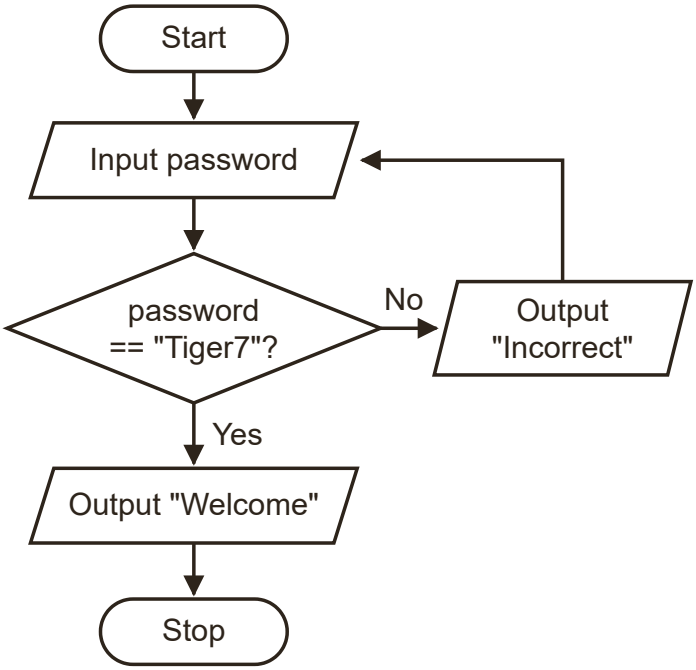
Notes

Flowchart symbols

- **Terminal** (rounded rectangle): Start or Stop.
- **Input/output** (parallelogram): data coming in or results going out, such as 'Input age' or 'Output total'.
- **Process** (rectangle): a calculation or assignment, such as 'total = total + price'.
- **Decision** (diamond): a question with two exits, labelled Yes and No (or True and False).
- **Sub program** (rectangle with a double line down each side): runs a separately defined sub program.
- **Lines with arrows** show the direction of flow. A loop is an arrow that goes back to an earlier step.



The flowchart symbols



A flowchart with a loop: the No arrow goes back to an earlier step

Pseudocode and Exam Reference Language

- **Pseudocode** is a way of writing an algorithm in structured, English-like steps. It does not follow the exact rules of any one language.
- Write one step per line and indent the lines inside loops and if statements, so the structure is clear.
- **Nesting** is putting one selection or loop inside another, such as an if statement inside a for loop. Indent each level further so it is clear which block each line belongs to.
- Questions in the exam show code in **OCR Exam Reference Language** (ERL). You need to be able to read it and write it.
- Example in ERL: a program that averages five numbers.

```
total = 0
for i = 1 to 5
   num = int(input("Enter a number"))
   total = total + num
next i
print("Average: " + str(total / 5))
```

Which form can you answer in?

- When a question just says 'Write an algorithm', you can usually answer in pseudocode, a flowchart, ERL or a high-level language.
- When a question says you must use OCR Exam Reference Language or a high-level programming language (as in Section B), flowcharts, pseudocode and English gain no marks.
- Small syntax slips are not usually penalised if the logic is clear, but missing logic always loses marks.

How to answer each type of question

Identify flowchart symbols

1 mark each | Grade 3

1. Match the shape to its job: rounded = terminal, parallelogram = input/output, rectangle = process, diamond = decision.
2. Give the name only; no description is needed unless asked.

Example. State the purpose of each flowchart symbol.



(a) (b) (c) (d)

(a) Diamond (b) Parallelogram (c) Rounded rectangle (d) Rectangle [4]

Model answer.

(a) Decision	(1)
(b) Input or output	(1)
(c) Terminal / start or stop	(1)
(d) Process	(1)

Complete an algorithm with gaps

2 to 4 marks | Grade 5

1. Read the description of what the algorithm should do.
2. Work out what each gap must be for that to happen: a condition, a variable, a value or an operator.
3. Check your answer by tracing it with one or two test values.

Example. The algorithm should keep asking for a password until `Tiger7` is entered, then output `Welcome`. Complete the gaps (1) to (3).

```
password = input("Enter password")
while password ....(1).... "Tiger7"
   print("Incorrect")
   password = ....(2)....
endwhile
print(....(3)....)
```

[3]

Model answer.

(1) `!=`	(1)
(2) `input("Enter password")`	(1)
(3) `"Welcome"`	(1)

Write an algorithm (Section A)

4 to 6 marks | **Grade 6**

1. List the requirements in the question: each one is usually worth a mark.
2. Set any totals or counters to 0 before a loop.
3. Put the inputs that repeat inside the loop, and the final output after it.
4. Indent the code inside loops and if statements.

Example. Write an algorithm that asks the user to enter the ages of 20 people, then outputs how many of them are 18 or over. [5]

Model answer.

```
count = 0
for i = 1 to 20
   age = int(input("Enter age"))
   if age >= 18 then
      count = count + 1
   endif
next i
print(count)
```

Loop that repeats 20 times (1); age input inside the loop (1); checks age `>= 18` (1); adds 1 to a counter that was set to 0 before the loop (1); outputs the counter after the loop (1).

Don’t lose marks

- Decision boxes with unlabelled exits, or with only one exit.
- Using a process rectangle for an input or output.
- Forgetting to set a total or counter to 0 before the loop.
- Putting the output inside the loop, so it is shown every time instead of once at the end.
- Answering in a flowchart or in English when the question demands ERL or a high-level language.

More tips

Memory tricks

- Flowchart shapes: Round = Run or stop, Parallel = Put in or out, Rectangle = do something, Diamond = Decide.
- Every decision diamond has one way in and two labelled ways out.
- Before writing code, jot down the inputs, the loop, the decision and the output: that is the skeleton of most answers.

Exam technique

- Look for the sentence 'You must use either OCR Exam Reference Language or a high-level programming language'. If it is there, write code.
- Use meaningful variable names and indentation: it makes your logic easy for the examiner to follow and credit.
- Trace your finished algorithm with a simple test value to check it does what the question asks.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3

Name the flowchart symbols

Terminal, input/output, process, decision and sub program symbols, joined by arrows.

Grade 4

Follow a flowchart or pseudocode to find outputs

Work through each step with given inputs, following the correct branch at each decision.

Grade 5

Complete missing parts of an algorithm

Fill gaps in pseudocode or a flowchart so that it meets the description.

Grade 6

Write an algorithm with selection and iteration

Write pseudocode or draw a flowchart that uses decisions and loops correctly.

Grade 7

Convert between a flowchart and code

Turn a flowchart into code, or code into a flowchart, keeping the same logic.

Grade 8

Refine an algorithm to meet new requirements

Change an existing algorithm, such as adding validation or a counter, without breaking it.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.