Paper 2 Section B practice: design, write, test and refine

Algorithms and programming (Paper 2) · revision notes

Section B of Paper 2 is worth 30 marks. Its questions follow one scenario and ask you to design, write, test and refine programs. Where a question says so, you must write your code in OCR Exam Reference Language or a high-level programming language you have studied: flowcharts, pseudocode and written English gain no marks there.

Key facts

- Section B = 30 marks: one scenario to design, write, test and refine
- Underline every requirement: about one mark each
- Cast numeric input: int(input("…"))
- Fixed number of repeats: for i = 1 to n … next i
- Test data: normal, boundary, invalid, erroneous, each with an expected result
- Code must be ERL or a high-level language when the question says so
- Initialise totals and counters (total = 0) before the loop
- Validation: do … until the input is valid
- Output results after the loop unless asked otherwise

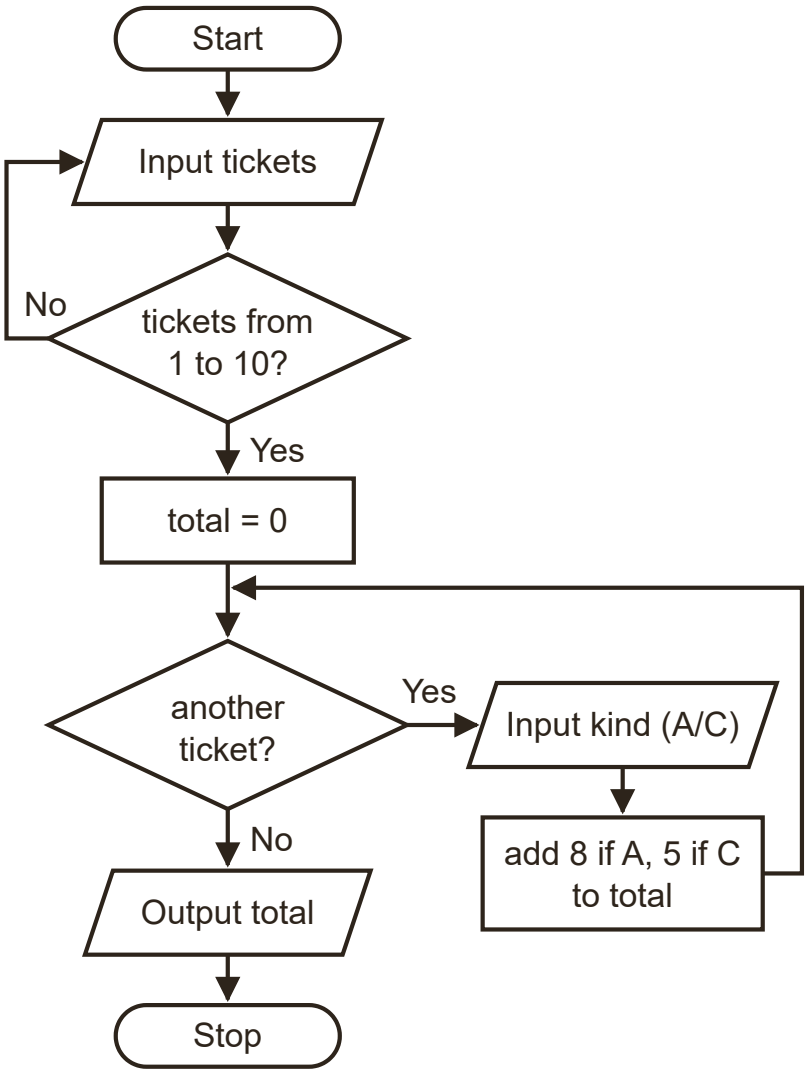
Notes

What Section B asks

- One scenario, such as a club, a shop or a game, runs through several linked parts.
- Typical parts: inputs, processes and outputs, or a validation check; completing or correcting code; writing a program; testing it; refining it to add a feature or fix a problem.
- The longer program-writing parts carry the most marks, with roughly one mark for each requirement met.
- Look for the sentence 'You must use either OCR Exam Reference Language or a high-level programming language that you have studied'. If it is there, only code gains marks.

Planning a program answer

- Underline every requirement in the question. Each one usually earns a mark.
- Decide the inputs, outputs and constructs needed: a loop? which type? selection? an array or a file?



A quick plan of the cinema program in the question below, before writing the code

- Set totals and counters to 0 before loops. Cast numeric inputs with int() or real() (float() in Python).
- Use the variable, array and file names the question gives.
- Indent your code, then tick off each requirement to check nothing is missing.

Techniques that come up again and again

- A validation loop: `do … until` the input is valid.
- A running total, count, largest or smallest value.
- Searching an array for a value, or looping through every element.
- Reading every line of a file, or writing lines to a file.
- A function with parameters that returns a value.
- String handling: length, substring, upper and concatenation.

Testing and refining

- Test plans need normal, boundary, invalid and erroneous data, each with the expected result.
- To refine code, state exactly which lines change and where new lines go, or rewrite the whole section.
- Trace your program with one test value before moving on.

How to answer each type of question

Write a program for the scenario 6 to 8 marks **Grade 7**

1. List the requirements from the bullet points.
2. Plan the order: validated input, loop, processing, output.
3. Write the code in ERL or a high-level language, indented, using the names given.
4. Tick off each requirement and trace with one simple test.

Example. A cinema charges £8 for an adult ticket and £5 for a child ticket. Write a program that:
• asks how many tickets are wanted, and keeps asking until a number from 1 to 10 is entered
• for each ticket, asks whether it is for an adult or a child (A or C)
• outputs the total cost.
You must use either OCR Exam Reference Language or a high-level programming language that you have studied. [6]

Model answer.

```
do
   tickets = int(input("How many tickets (1-10)?"))
until tickets >= 1 AND tickets <= 10
total = 0
for i = 1 to tickets
   kind = input("Adult or child (A/C)?")
   if kind == "A" then
      total = total + 8
   elseif kind == "C" then
      total = total + 5
   endif
next i
print("Total cost: £" + str(total))
```

Inputs the number of tickets as an integer (1); repeats the input until it is from 1 to 10 (1); a loop that runs once for each ticket (1); inputs the ticket type inside the loop (1); adds 8 for A and 5 for C to a total set to 0 before the loop (1); outputs the total after the loop (1).

Complete a test plan for the program

3 to 4 marks **Grade 5**

1. Use the rule from the scenario (here, 1 to 10 tickets).
2. Choose one value for each type of test asked for.
3. Give the expected result: accepted, or rejected and asked again.

Example. Complete a test plan for the input of the number of tickets in the cinema program, with one normal, one boundary, one invalid and one erroneous test. [4]

Model answer.

Type of test	Test data	Expected result
Normal	4	Accepted (1)
Boundary	10 (or 1)	Accepted (1)
Invalid	11 (or 0)	Rejected and asked again (1)
Erroneous	"six"	Rejected and asked again (1)

(The program above would need a type check added to pass the erroneous test, because int() cannot convert "six".)

Identify and fix an error in given code

2 to 3 marks **Grade 6**

1. Trace the code with a small value to see what really happens.
2. Find the line that causes the difference.
3. Write the corrected line in full.

Example. A student wrote this line for the ticket loop: `for i = 0 to tickets`
Explain the problem and write a corrected line. [2]

Model answer. The loop runs tickets + 1 times, so the program asks about one ticket too many (1).
Corrected: `for i = 1 to tickets` (or `for i = 0 to tickets - 1`) (1).

Refine the program 3 marks **Grade 9**

1. Decide what new code is needed and exactly where it must go.
2. Write the new lines in full.
3. Check the change does not stop the rest of the program working.

Example. The cinema now takes £2 off the total cost when 5 or more tickets are bought. Write the extra code needed and state where it goes in the program. [3]

Model answer.

```
if tickets >= 5 then
   total = total - 2
endif
```

Condition tickets >= 5 (1); subtracts 2 from the total (1); placed after the loop and before the total is output (1).

Don't lose marks

- Answering a code question in pseudocode, a flowchart or English when ERL or a high-level language is required.
- Forgetting to cast input to an integer, so comparisons and arithmetic go wrong.
- Validating with an if statement only, so an invalid value is never asked for again.
- Setting the total to 0 inside the loop, so it resets every time.
- Putting the output inside the loop, or missing a requirement from the list.

More tips

Memory tricks

- Before writing, make a mini-plan: Inputs, Loop, Decide, Output (ILDO).
- Each bullet point in the question is a checklist item: tick it off in your code.
- Trace with the smallest valid input (such as 1 ticket) to catch loop errors.

Exam technique

- Read the whole scenario first: later parts often reuse the variables and code from earlier ones.
- Small syntax slips are not usually penalised if the logic is clear, so focus on getting the logic right.
- Spend your time in proportion to the marks; if stuck, write the parts you can (inputs, loop, output) to pick up marks.

What each grade needs

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

Grade 4 **Identify inputs, processes and outputs for a scenario**
Pick out the data in, what is done with it and what comes out.

Grade 5

Complete a test plan for a scenario

Choose normal, boundary, invalid and erroneous data with expected results.

Grade 5

Complete or correct lines of given code

Fill gaps or fix errors so the code does what the scenario needs.

Grade 6

Write a program with input, selection and iteration

Combine loops, if statements and variables to meet every requirement.

Grade 7

Validate input inside a loop

Keep asking until the input meets the rule, before using it.

Grade 8

Use arrays, files or sub programs in solutions

Store and process data with the right structure, such as an array of names.

Grade 9

Refine a program to meet new requirements

Change working code accurately, without breaking what it already does.

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.