A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

9.2.3 Structured English, stepwise refinement and describing an algorithm

Name: _____ Class: _____ Date: _____ **Total: 48 marks**

KEY WORDS structured English 结构化英语 • stepwise refinement 逐步求精 • algorithm 算法 • pseudocode 伪代码 • flowchart 流程图

Objective

Learn to write an algorithm **in words**: as **structured English** 结构化英语, as the numbered steps of a **stepwise refinement** 逐步求精, and as a description with no pseudocode in it. You also turn structured English into pseudocode. This is the last sheet of 9.2: flowcharts were on sheet 9.2.2, and the loops you need are on sheets 11.2.2 and 11.2.3.

You must be able to:

- document a simple algorithm as a structured English description
- write pseudocode from a structured English description
- describe stepwise refinement, and use it to break a step down until it can be programmed
- describe an algorithm in words, without pseudocode statements

1 Worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

■ One algorithm, three ways

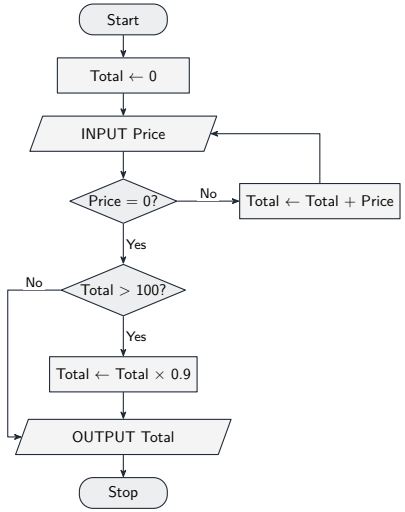
A shop till adds up the prices of the items in a basket. The cashier enters 0 after the last item. If the total is more than 100, 10% is taken off. The till outputs the amount to pay.

The same algorithm can be written in three ways, and you must be able to move between them.

Structured English: short numbered sentences with the command words of programming (*input, output, set, if, repeat*), but no strict rules.

1. Set the total to 0.
2. Input the price of an item.
3. If the price is 0, go to step 6.
4. Add the price to the total.
5. Go back to step 2.
6. If the total is more than 100, take 10% off the total.
7. Output the total.

Flowchart



Pseudocode

```
DECLARE Price, Total : REAL
Total ← 0
INPUT Price
WHILE Price <> 0
    Total ← Total + Price
    INPUT Price
ENDWHILE
IF Total > 100 THEN
    Total ← Total * 0.9
ENDIF
OUTPUT Total
```

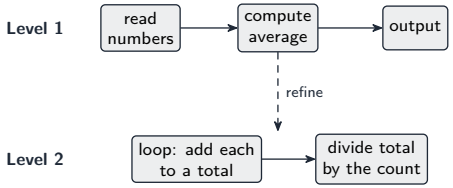
■ From structured English to pseudocode

1. List the data the steps use, and declare each item.
2. Take the steps **in order**. "Set …" and "add …" become assignments with ←; "input" and "output" keep their names.
3. "If …" becomes `IF … THEN … ENDIF`.
4. "Go back to step …" or "repeat … until …" is a **loop**: test **before** the value is used → `WHILE`; run once, then test → `REPEAT … UNTIL`.

5. Do not copy "go to step 6" into the pseudocode: the loop replaces it.

■ Stepwise refinement

Stepwise refinement: write the task as a few large steps, then **break each step down into smaller steps**, until **every step is simple enough to program**.



Task: produce a grade report for a class test.

1. Input the marks.
   - 1.1 For each of the 30 students, input the name and the mark.
   - 1.2 If a mark is not from 0 to 50, ask for it again.
2. Work out each grade.
   - 2.1 If the mark is 40 or more, the grade is A.
   - 2.2 Otherwise, if the mark is 25 or more, the grade is B.
   - 2.3 Otherwise, the grade is C.
3. Output the report.
   - 3.1 Output each name with its grade.
   - 3.2 Output how many students got each grade.

- Steps 1 to 3 say **what** must be done. Steps 1.1 to 3.2 say **how**, in enough detail to program each one.
- Stop refining when one step matches about **one statement** of pseudocode.

■ Describing an algorithm in words

When a question says **describe** the algorithm and **do not include pseudocode statements**, write short numbered sentences. For the till:

1. Set the total to zero.
2. Input the price of the first item.
3. Repeat steps 4 and 5 until the price input is zero.
4. Add the price to the total.
5. Input the price of the next item.
6. If the total is more than 100, reduce it by 10%.
7. Output the total.

A full answer names **five** things: the values **set first**, what the **loop repeats**, **when it stops**, each **decision**, and the **output**. Leave out pseudocode words such as `WHILE`, `ENDIF` and `←`.

2 Practice

Now use the methods above. The first questions need one idea each.

2.1 The steps of an algorithm in structured English are in the wrong order. The algorithm inputs ten numbers and outputs how many of them are negative.

- **A** Output the count.
- **B** Repeat the next two steps ten times.
- **C** Set the count to 0.
- **D** If the number is less than 0, add 1 to the count.
- **E** Input a number.

Write the letters in the correct order. [3]

2.2 Describe what is meant by **stepwise refinement**. [2]

2.3 State how you know when to **stop** refining a step. [1]

2.4 A student describes an algorithm in words, but the question said "do not include pseudocode statements". Rewrite each line as a sentence in plain English. [3]

(a) `Count ← Count + 1`

(b) `WHILE Mark <> -1`

(c) `OUTPUT Total / Count`

2.5 Write pseudocode from this structured English. Declare your variables. [5]

1. Set `Count` to 0.
2. Input a word.
3. If the word has more than 5 characters, add 1 to `Count`.
4. Repeat from step 2 until the word `"STOP"` is input.
5. Output `Count`.

The function `LENGTH()` returns the number of characters in a string.

2.6 The task *work out the fare for a taxi journey* has three steps: input the distance and the time; calculate the fare; output the fare. The fare is 3.00, plus 1.50 for each km, and the whole fare is doubled from midnight to 6 a.m. Use stepwise refinement to break the step **calculate the fare** into smaller steps. Do not use pseudocode. [4]

3 Exam-style questions

3.1 A sports club needs an algorithm that inputs the ages of people who want to join. The number of people is not known, and the value 0 is input after the last age. The algorithm outputs how many of the people are under 18.

Write the algorithm in **structured English**, as numbered steps. [5]

3.2 A garage records the mileage of each car it services in a day. The number of cars is not known in advance, and 0 is entered after the last car. The algorithm must output the mean mileage and the number of cars with a mileage of more than 100 000.

Describe the algorithm. Do **not** include pseudocode statements in your answer. [5]

3.3 An algorithm is described in structured English:

1. Input a temperature.
2. If the temperature is -999, go to step 6.
3. If the temperature is higher than the highest so far, it becomes the highest so far.
4. Add 1 to the number of readings.
5. Go back to step 1.
6. Output the highest temperature and the number of readings.

Before step 1, the highest temperature is set to -100 and the number of readings is set to 0.

Write pseudocode for this algorithm. Declare your variables. [6]

3.4 A program produces an invoice for a customer's order. At the first level the algorithm has three steps: input the order; calculate the invoice total; output the invoice.

Each item in the order has a price and a quantity. A delivery charge of 5.00 is added if the goods cost less than 50.00, and tax of 20% is then added to the whole amount. Use stepwise refinement to break the step **calculate the invoice total** into at least **four** smaller steps. Do not use pseudocode. [5]

3.5 A teacher inputs the exam marks of a class, one mark at a time. The value -1 is input after the last mark. An algorithm must output how many marks are a pass (50 or more), and the highest mark.

(a) Stepwise refinement is applied to the algorithm. Describe up to **six** steps for this algorithm that could be used to produce pseudocode. Do **not** use pseudocode statements in your answer. [6]

(b) The teacher now also wants the **mean** mark. State the changes that are needed to your steps. [3]

Solutions

Each answer shows the steps, then the **result**.

2.1

- the count is set first, and the loop comes next: C, B
- inside the loop a number is input and then tested: E, D
- the output comes after the loop: **C, B, E, D, A**

2.2

- a task is written as a few large steps, and each step is broken down into smaller steps
- this is repeated until every step is simple enough to be programmed

2.3

- when each step is so simple that it can be written as about one statement of program code

2.4

(a) add 1 to the count

(b) repeat (the following steps) while the mark is not -1

(c) output the total divided by the count

2.5

- declarations and `Count ← 0`
- a post-condition loop that ends when the word is `"STOP"`
- input the word inside the loop
- `IF LENGTH(Word) > 5` adds 1 to `Count`
- output `Count` after the loop

```
DECLARE Word : STRING
DECLARE Count : INTEGER
Count ← 0
REPEAT
    INPUT Word
    IF LENGTH(Word) > 5 THEN
        Count ← Count + 1
    ENDIF
UNTIL Word = "STOP"
OUTPUT Count
```

2.6 any **four** of:

- set the fare to 3.00
- multiply the distance in km by 1.50
- add this amount to the fare
- check whether the time is from midnight to 6 a.m.

- if it is, multiply the fare by 2

3.1 one mark for each point, in a sensible order:

1. Set the count to 0.
2. Input an age.
3. If the age is 0, go to step 6.
4. If the age is less than 18, add 1 to the count.
5. Go back to step 2.
6. Output the count.

3.2 any **five** of:

- set the total mileage, the number of cars and the number of high-mileage cars to zero
- input the mileage of the first car
- repeat until the mileage input is zero
- add each mileage to the total, and add 1 to the number of cars
- if a mileage is more than 100 000, add 1 to the number of high-mileage cars
- input the next mileage at the end of each pass
- after the loop, output the total divided by the number of cars, and the number of high-mileage cars

3.3

- the variables declared, and the two starting values set
- a first input **before** the loop
- a pre-condition loop that continues while the temperature is not -999
- the `IF` that updates the highest temperature
- 1 added to the number of readings, and the next input, inside the loop
- both outputs after the loop

```
DECLARE Temp : REAL
DECLARE Highest : REAL
DECLARE Readings : INTEGER
Highest ← -100
Readings ← 0
INPUT Temp
WHILE Temp <> -999
    IF Temp > Highest THEN
        Highest ← Temp
    ENDIF
    Readings ← Readings + 1
    INPUT Temp
ENDWHILE
OUTPUT Highest, Readings
```

- the value is tested before it is used (step 2 comes before steps 3 and 4), so this is a `WHILE` loop; a `REPEAT` loop would count the -999 as a reading

3.4 any **five** of, in a sensible order:

- set the goods total to 0
- for each item, multiply the price by the quantity
- add each result to the goods total
- if the goods total is less than 50.00, add 5.00 for delivery
- calculate the tax as 20% of this amount
- add the tax to give the invoice total

3.5

(a) one mark for each step:

1. Set the number of passes to 0 and the highest mark to 0.
2. Input the first mark.
3. Repeat steps 4 to 6 until the mark input is -1.
4. If the mark is 50 or more, add 1 to the number of passes.
5. If the mark is greater than the highest mark, set the highest mark to this mark.
6. Input the next mark.
7. Output the number of passes and the highest mark.

(b) set a total and a count of marks to 0 in step 1

- inside the loop, add each mark to the total and add 1 to the count of marks
- after the loop, output the total divided by the count of marks