Y10 Computer Science summer support booklet 2

Algorithms and Programming

Contents

Algorithms 4
Understanding Algorithms 4
Flow Charts 4
Pseudocode 6
Pseudocode Average of N Numbers Algorithm 7
Detecting and Correcting Errors in Algorithms 7
Trace tables 8
Programming Languages 9
High- and Low-Level Languages 9
Programming constructs 11
Python challenges 11
Programming questions 12
Algorithms 13
Programming languages 16
Programming constructs 17
Handling data in algorithms 18

Algorithms

Understanding Algorithms

Computer science-type problems are solved using algorithms. An **algorithm** is a step-by-step approach to solving a problem. It is written in pseudocode, which is independent of any particular programming language. **Flow charts** or flow diagrams use a series of symbols and arrows to indicate the flow of information.

Algorithms can be used to describe non-computer-type problems as shown in the travel example below. Once a problem has been specified, a series of steps or an algorithm can be created to produce a result.

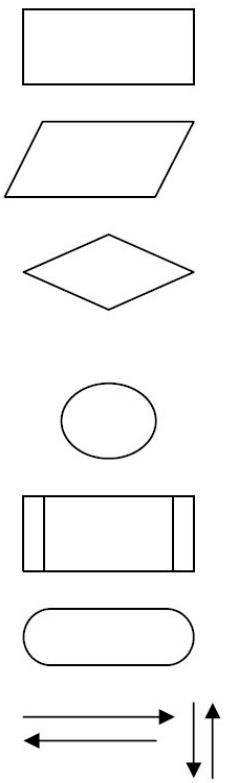
Problem – Need to Travel to New York

1. Contact travel agent and purchase ticket
2. collect travel documents
3. pack luggage
4. travel to airport
5. board correct plane
6. collect luggage
7. use taxi to leave airport

Obviously, the model could be refined to add further steps, such as choosing different routes, dates and price options.

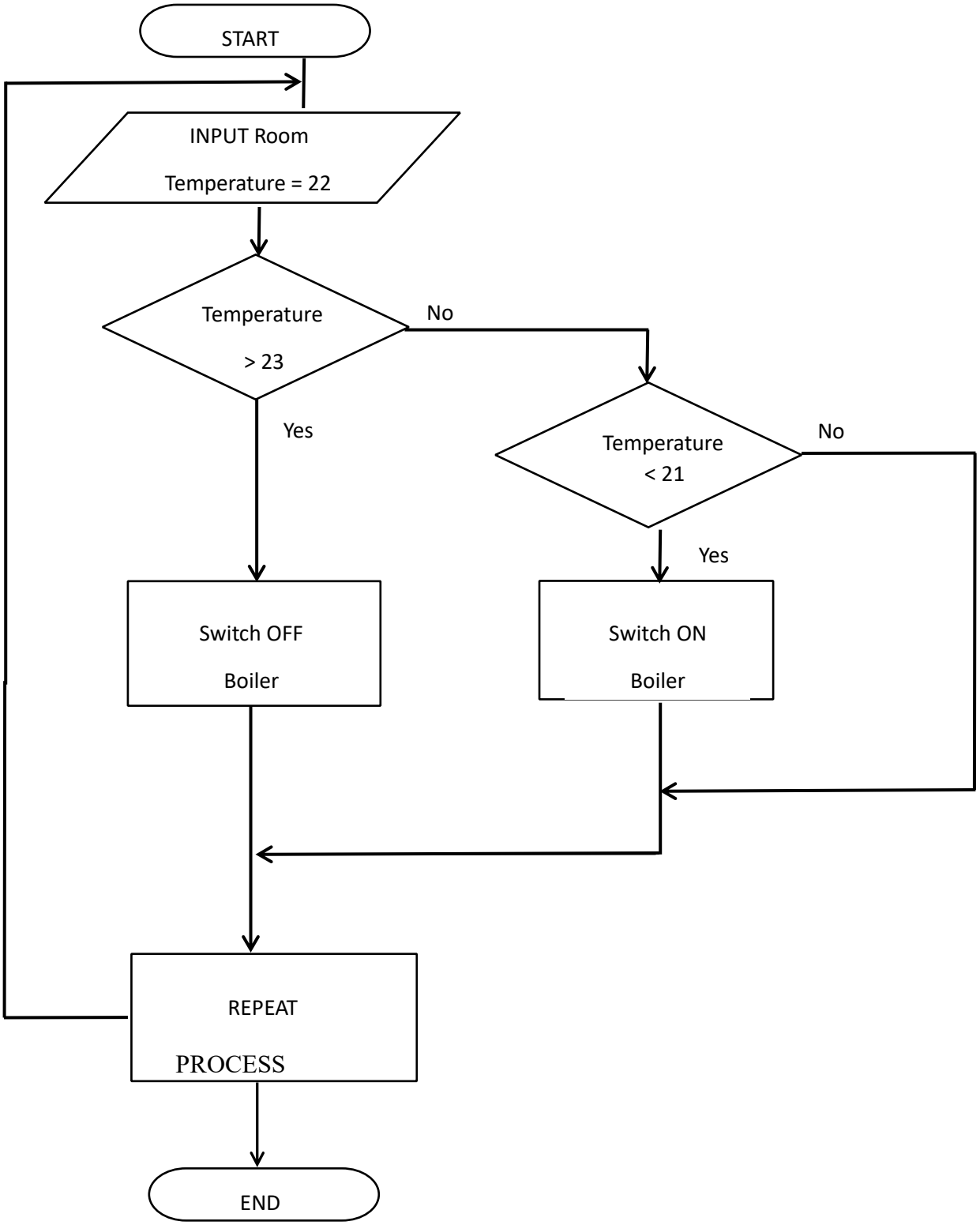
Flow Charts

The following symbols are used in flow chart diagrams to show the steps through an algorithm or program.



Used to describe a process or operation in the program
Used to indicate any input or output of data in the program
In a decision diamond, there will be an input and two outputs where one represents true and the other false
Connecting circles are used to link parts of a flow chart that are drawn on more than one page
This symbol is used to indicate that a pre-defined subroutine has been used.

Flow Chart Example – Use a flow chart to indicate how temperature is controlled in a central heating system.



In this case, the required temperature is selected as 22.

When the room temperature is higher than 22, the boiler is turned off.

When the room temperature is lower than 22, the boiler is turned on.

Flow charts are excellent at giving a pictorial and easy-to-understand representation of a solution, but they are seldom used commercially for presenting algorithms as they need special software to create them and they are difficult to modify. Consequently, **pseudocode** is now widely used; the main advantages and disadvantages of using pseudocode are tabulated below:

Advantages of Pseudocode	Advantages of Flow Charts
Pseudocode can be easily created and modified using a word processor.	Flow charts give a pictorial representation of a problem which aids understanding.
It can also make use of indentation to clearly display structured concepts.	The symbols used are universal and so they are neither programming nor natural language dependent.

Pseudocode

An alternative method to using a flow chart is to use pseudocode to describe an algorithm.

There are no hard and fast rules for writing pseudocode, but algorithms written in pseudocode are easier to interpret when written using the following ideas:

Pseudocode is an outline of code or a complete program that can easily be converted to the particular programming language being used.

- Capitalise the keywords: such as INPUT, OUPUT, IF, THEN, ELSE, END, REPEAT, UNTIL.
- Write one statement or action on each line.
- Use meaningful variable names where possible.
- Use indentations of statements to show structure clearly, for example:

```
IF a > 1
      b ← 5
ELSE
      b ← 1
END
```

Note – indentation is used in the average numbers example.

Avoid programming language special features as the pseudocode should be language independent.

Pseudocode Average of N Numbers Algorithm

Example: Write an algorithm in pseudocode to find the total and average of N numbers, until a negative number is input.

The following steps take place:

1. Initially the variables average, count and total are zeroed.
2. The main calculations are carried out within the **REPEAT... UNTIL** N < 0 loop (in many programming languages this would be a DO… WHILE loop).
3. The first time through the loop, a number N is inputted, and the variable count is used to keep track of the numbers inputted.
4. Next, the total and average are calculated, and a new value of N is inputted.
5. When the value of N inputted is less than zero, the sequence stops and the count, total and average are outputted.

Average of N Numbers Algorithm

```
Average = 0
Count = 0
Total = 0
REPEAT
        INPUT N
        Add 1 to Count
        Total = Total + N
        Average = Total / Count
UNTIL N < 0
OUTPUT  Count,  Total,
Average
```

Meaningful variable names are used to aid the interpretation of the function.

Detecting and Correcting Errors in Algorithms

Errors can be detected and corrected in algorithms by using a trace table. The example below traces the average of N numbers algorithm to ensure that the pseudocode algorithm gives the expected results.

Number	Count	Total
	0	0
4	1	4
8	2	12
12	3	24

A **trace table** is a step-by-step method of testing an algorithm. Each row of the table shows the numbers input and the results produced. Any errors detected can then be corrected.

It can be seen from the trace table that the results are as expected. There could have been errors if the calculations had been carried out in a different order.

Trace tables

Complete the following trace tables to trace through the algorithms.

```
num = 3
n = 0
while n > 4
    num = num + n
endwhile
output (num)
```

num	n	N < 4	output
3	0	TRUE	

Assume the user enters: 8, 3, 6, 5, 10, 6

```
total = 0
for count = 1 to 3
   base = input
   height = input
   x = (base * height)/2
   total = total + x
next count
result = total/3
print (result)
```

total	count	base	height	x	output
0	1	8	3	12	

Programming Languages

Machine code is the set of binary instructions that are used by the CPU to perform a task.

High- and Low-Level Languages

Machine code is not readily understandable as it is written in long binary strings. Machine language is dependent upon the processor that is using it; in each case, the processor reads a binary instruction and interprets it to perform different simple operations such as accessing or storing data in RAM; therefore, carrying out an operation such as adding two different numbers would take many machine code instructions.

The following examples of machine code give an indication of how difficult it is to be read and understood by programmers.

In machine code, the registers are used by the processor to store data that is being processed.

Load data into register 8, taken from memory cell 68 after location listed in register 3				
Operation	**Register Operations**		**Memory Address**	
35	3	8	68	denary
100011	00011	01000	00000 00001 000100	binary

Jumping to address 1024		
Operation	**Target Memory Address**	
2	1024	denary
000010	00000 00000 00000 10000 000000	binary

Note that it is not necessary to learn any machine code for this course, but simply to have an awareness that the CPU only runs machine code and that it is very difficult to read and understand machine code.

Programming in machine code is not a practical option as it would be very time-consuming.

In some cases, programmers will write programs in assembly language, which is a low-level computer programming language but more understandable than machine code. The preferred method of writing program is using high-level languages.

This is mainly because they are easy to understand and written using a natural language notation, making them easier to maintain.

Assembly language is a low-level computer programming language which is processor-dependent where each instruction is one machine operation.

High-level language is a computer programming language based on natural language or mathematical notation.

Assembly Language Example

```
SET r1, 12 ;    set register 1 to 12
STORE A, r1 ;    store register 1 contents into variable A
```

C++	Visual Basic	Pascal	Python
int A, B; A = 12; B = A;	Dim A,B As Integer A = 12 B = A	VAR A, B : INTEGER; BEGIN A := 12; B := A; END	A =12 B =A

The high-level language examples are similar and easy to understand when compared to machine code in binary or even the assembler code shown above.

It is necessary for programs written in high-level code or even assembler code to be converted to machine code so that the processor can understand and carry out the instructions; this process is carried out using a software program known as a translator.

Translation is carried out using one of the following software programs:

- **Assembler** (used for Assembly Language programs)
- **Compiler** (used for program languages such as: C++, Visual Basic and Pascal)
- **Interpreter** (used for program languages such as: some versions of Basic and Python)

Translator	Characteristics of the Programming Languages and their Languages
Assemblers	The source code is written in assembly language, which is a series of memorable mnemonics used to represent machine operational codes. The assembler translates this source code into machine code that the computer can run. The advantage of using assembly language is it is an efficient low-level language that can be translated quickly as it has a one-to-one relationship with machine code. The main disadvantage of using an assembly programming language is that it is complex to write and takes lots of programing time and expertise.
Compilers	The source code created by the programmer is not understandable by the computer. A software application called a compiler converts the source code to object code, which the computer can run. The object code is converted to match the target computer that will run the software. The main advantage of using a compiler on high-level language code is that an executable file is produced which runs without the need of the source code; this makes the source code more secure as it does not need to be distributed to the customer. The disadvantages of this method are that the compilation of a large program takes a long time to complete and any errors in the source code need to be corrected before an executable file can be produced; so, if there are any errors, it is necessary to correct them and recompile the source code.

Interpreters	Interpreter software normally executes the source code directly; this avoids the need to compile the program. Interpreted software runs more slowly than compiled software as statement in the program is analysed before it is executed. The advantage of using an interpreter is that during development the programmer might make frequent changes, which can be tested without going through the time-consuming process of compiling and linking for each change. The disadvantage of using an interpreter is that it needs to be loaded on the target computer, where it creates efficient machine code at runtime. Additionally, the source code is available to the customer and the line-by-line translation method takes longer than a compiled program to run.

Programming constructs

Python is a high-level language and the language you need to understand well for the Computer Science exam. There are three main constructs in programming that all programming languages are based on:

Sequence – All programs have a series of steps to be followed in sequence
Selection – choice of path through the program (if, then, else)
Iteration – repeated steps (looping) in the program (for loop, while loop)

Python challenges

Use the website - https://create.withcode.uk/ to write the code.

Alternatively you can use Python Visualiser - https://pythontutor.com/visualize.html#mode=edit

1. Write a program for working out the wages for a paperboy or papergirl. The program should ask how many papers were delivered each day for 7 days and then display the total amount earned. The newsagent pays 10p per newspaper delivered.

   Hint: *Need a For loop (count-controlled)*

2. Write a program for calculating the bill in a restaurant. The program should keep asking for the price of another item until the user enters 0. It should then print out the total bill.

   Hint: *Need a while loop*

3. Write a program to calculate the total number of goals scored in a tournament. First, the program should ask how many matches were played. For each match, the program should ask how many goals were scored. Once all the data has been entered the program should display the total number of goals scored and the average goals per match.

Programming questions

Programming: Questions

1 (a) Explain the term 'algorithm'. (2 marks)

.....

.....

(b) Using pseudocode, write a simple algorithm to add two numbers. (4 marks)

.....

.....

.....

.....

(c) List one advantage and one disadvantage of using pseudocode to create an algorithm. (2 marks)

..... Advantage:
.....

.....Disadvantage:
.....

2 (a) Computer programmers often use structured programming techniques when creating gaming software. Name two advantages of adopting this approach. (2 marks)

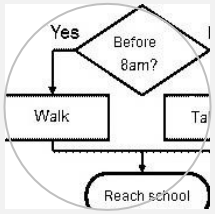
..... Advantage 1:
.....

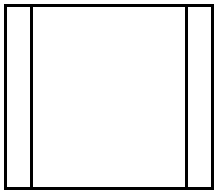
..... Advantage 2:

(b) Explain the reasons why software problems are broken down into smaller problems when creating a solution. (2 marks)

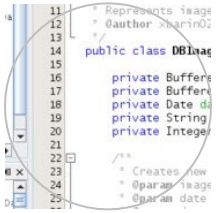
.....

.....

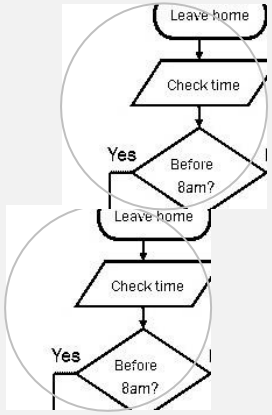
Algorithms  	What is an algorithm?	
	Who is regarded as the first ever computer programmer?	
	What was the name of the first ever general-purpose, high-level programming language?	
	State and describe two different ways in which an algorithm can be represented.	***1.*** ***2.***
	What shape on a flow chart is used to represent a selection (decision or choice)?	
	What shape on a flow chart is used to represent an input or output?	

	What does this flow chart symbol represent? 	
	What is meant by 'tracing' an algorithm?	
	Draw a flow chart for an algorithm to represent the following situation: *A boy leaves his house in the morning to travel to school.* *He checks the time on his watch when he sets off. If the time is before 8am, he decides to walk to school.* *However, if the time is 8am or later he decides to take the bus.* *He always follows this rule, and either way he will reach school on time.*	

	Ensure that you use the correct symbols for each stage of the algorithm.	
	Write an equivalent algorithm in pseudocode for the same process.	

Programming languages 	What is machine code?	
	What is assembly language? How did it evolve from machine code?	
	What defines high-level languages?	
	Describe the function of translator software.	
	Describe each of the following translators:	***Assembler:***
		Interpreter:
		Compiler:
	What is an IDE?	
	Describe each of the following tools that might be provided by an IDE.	***Code editor:***
		Error diagnostics:

		Run-time environment:
		Auto-documentation:

Programming constructs 	What is meant by 'sequence' in programming?		
	What is meant by 'selection'?		
	Give two examples of programming constructs which use selection.	***1.***	***2.***
	What is meant by 'iteration'?		
	Give two examples of programming constructs which use iteration.	***1.***	***2.***

	Read the following pseudocode example. Identify an instance of sequence, selection and iteration.	START program INPUT A INPUT B FOR j = 1 to 10 C = (A +B)/2 IF A = B - 4 A = 6 B = 13 END IF NEXT j OUTPUT C END program ***Sequence:*** ***Selection:*** ***Iteration:***

Handling data in algorithms	Explain what is stored by each of the following data types:	***Integer:***
		Real/float:
		Boolean:
		Character:
		String:

	Suggest an appropriate data type to store each of the following data items.	

The quick brown fox jumps over the lazy dog	
A	
872.15	
54	
TRUE	

Explain the difference between variables and constants.	
Explain the difference between declaration and assignment.	
What is an array used for?	
What is meant by the dimension of an array?	