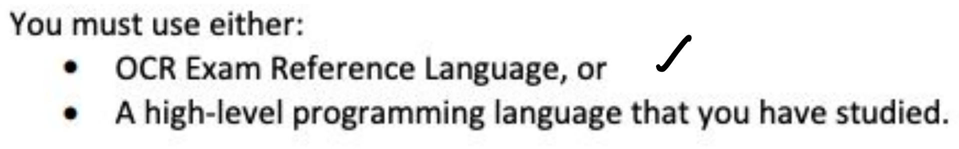
OCR Paper 2 Live Stream

Write in pseudocode unless it has this



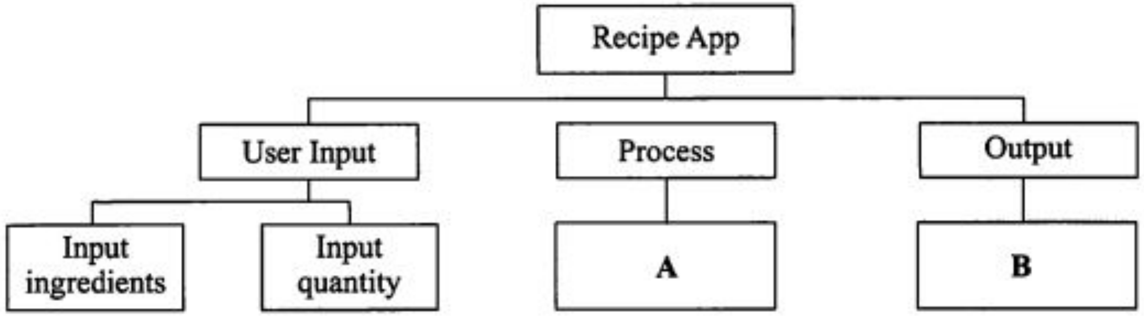
You must use either:

- OCR Exam Reference Language, or ✓
- A high-level programming language that you have studied. ✓

2.1.1 Computational Thinking

- **Decomposition** is breaking a problem down into smaller sub-problems. Once each sub-problem is small and simple enough it can be tackled individually.
- **Abstraction** is removing or hiding unnecessary details from a problem so that the important details can be focused on or more easily understood.
- ***Algorithmic thinking** involves deciding on the order that instructions are carried out and identifying decisions that need to be made by the computer

Sophia wants to develop a recipe app for her smartphone. The app will ask a user to input ingredients and will return a list of recipes that can be made. Part of a structure diagram for the app is shown below.



Suggest a module or task that would be appropriate for Box A and Box B.

Box A: Search the database

Box B: Display recipe

[2]

Which technique of computational thinking is shown with a structure diagram?

Decomposition.

[1]

1. (a) Tick one box in each row to identify whether the statement is an example of abstraction or decomposition.

Statement	**Abstraction**	**Decomposition**
Breaking down a complex problem into smaller, more manageable parts		✓
A web developer is designing a website. They create reusable components such as buttons and cards.		✓
Representing characters in the game as simple geometric shapes instead of detailed human models	✓	

[3]

a) State the value of 2 ^ 4 16

(b) State the value of 13 DIV 4

3
____ [1]

(c) A programmer declares the following variables.

first = "The GCSE Computer Science Tutor" – 31
second = "is the best" – 11
State the output from the following lines of program code.

(i) print(first.substring(20,5) ience ______________________ [1]

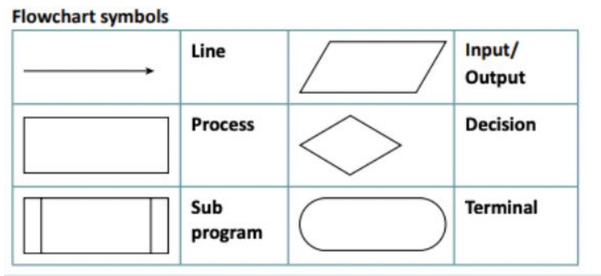
(ii) print(first.length MOD second.length) 9 ______________________ [1]

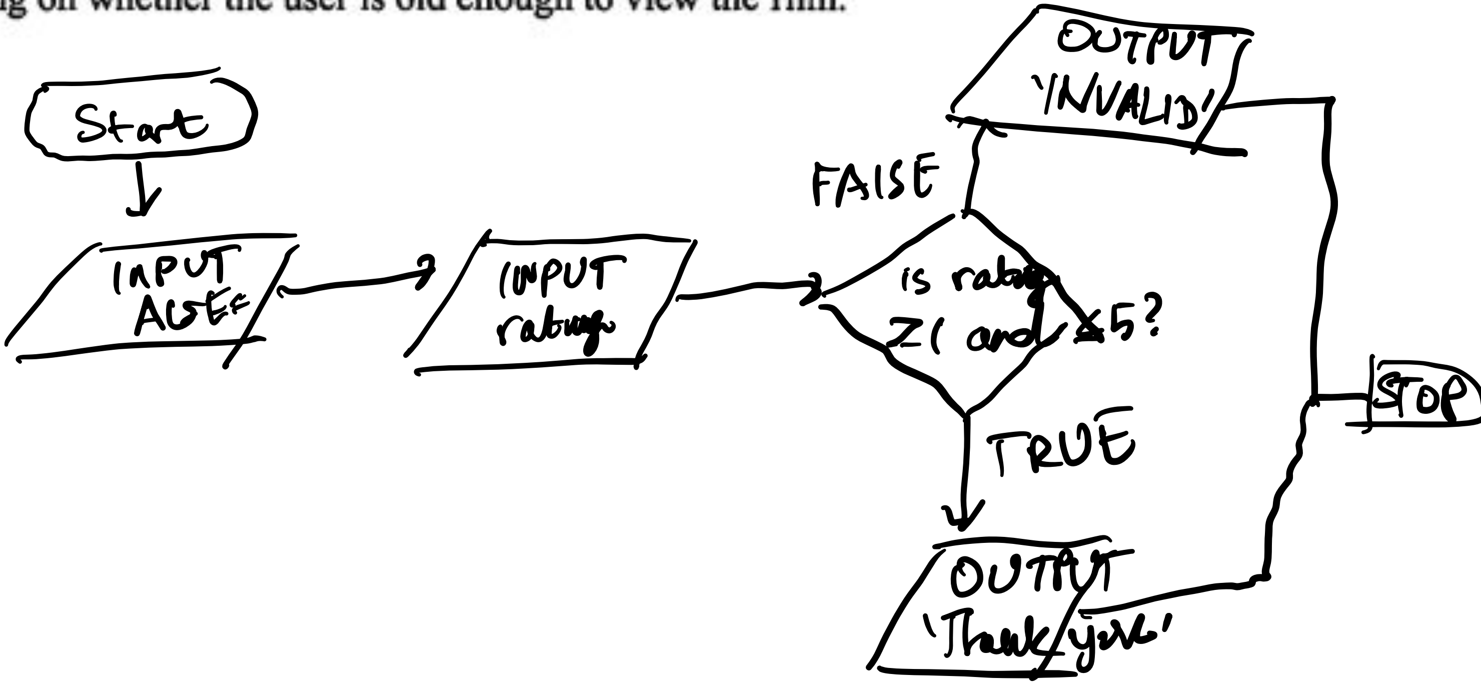
↓
remainder

Flowcharts

A film streaming website uses an algorithm to restrict which films a user can view.
Draw a flowchart to show this algorithm. Your flowchart should:

- Have one input for the user's age and one input for the age rating of the film.
- Allow or deny access depending on whether the user is old enough to view the film.





Trace Table

A trace table is a **tool** that can be used to **follow each line of an algorithm** through, step by step. The trace table will show the **contents of each variable** after each line has been carried out and will also show any output.

By manually following through the algorithm in this way, we can see if it produces the correct result and if not, help us to identify where any **errors** have occurred.

(d) Read the following pseudocode algorithm:

```
01  start = 3
02  do
03    print(start)
04    start = start - 1
05  until start == -1   ⊖ ✓
06  print("Finished")
```

Complete the following trace table for the given algorithm.

Line	Start	Output
01	3	
03		3
04	2	
03		2
04	1	
03		1
04	0	
03		0
04	-1	
06		Finished

Bubble Sort

Bubble Sort:

- The bubble sort algorithm works by comparing pairs of values.
- If the two values are in the wrong order with respect to each other, they are swapped over.
- This is then repeated for each further pair of values. When the last pair of values has been compared, the first pass of the algorithm is complete.
- The algorithm will repeat until a pass has been completed with no swaps occurring.
- Once this happens, the list is guaranteed to be in order.
- After the first pass the highest element in the list bubbles towards the end

Pros

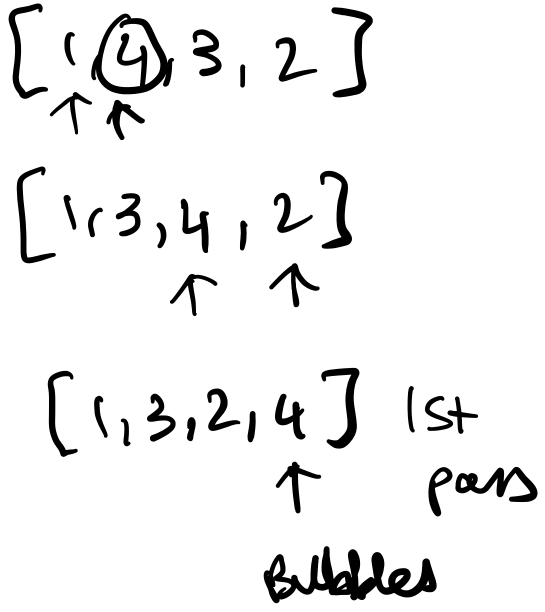
- Easy to implement ✓
- Does not use much memory ✓

Cons

- Poor for efficiency, especially if the list is scrambled up. It would take too many iterations to sort the list.

(b) Trudi has written a procedure, `bubbleSort`.

```
procedure bubbleSort(numbers)
      do
            sorted = true
            for count = 0 to numbers.length -2
                  if numbers[count] > numbers[count+1] then
                        temp = numbers[count+1]
                        numbers[count+1] = numbers[count]
                        numbers[count] = temp
                        sorted = false
                  endif
            next count
      until sorted == true
endprocedure
```



(iii) Describe the purpose of the `temp` variable in the procedure `bubbleSort`.

Data Stored temporarily
– to allow the contents of
two Variables to be swapped.

[2]

(iv) Describe the purpose of the `sorted` variable in the procedure `bubbleSort`.

used to indicate weather a
swap is made or not

[2]

(c) The target integer 8 exists in a list of integers 1, 4, 6, 9, 8, 12, 15 but is not found during a binary search. There are no errors in the code.

(i) Give the reason why the target integer 8 is **not** found.

List is not sorted!

[1]

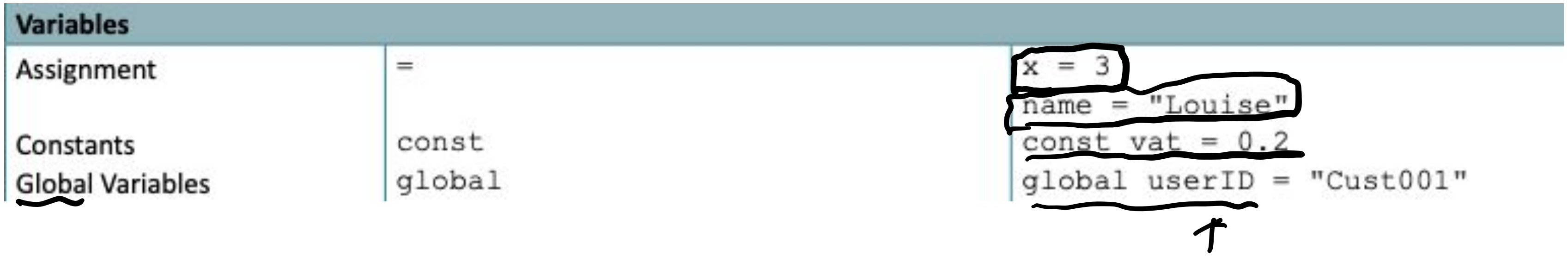
(ii) Identify and describe an alternative search algorithm that could be used.

Linear search — each element is compared to the item we want to find.

[3]

Variable - stores a single piece of data. It is a label for an allocated area of memory. The value of a variable can be changed during the execution of the program.

Constant - is also a label for an allocated area of memory. Unlike a variable, the value of a constant cannot change during the execution of the program.



Variables		

Assignment	=	`x = 3`
		`name = "Louise"`

Constants	const	`const vat = 0.2`
Global Variables	global	`global userID = "Cust001"`

Programming Constructs – Building blocks programmers use to write code. There are 3 main types:

Sequence - is the execution of statements one after the other, in order. A program runs from top to bottom and each instruction completes fully before the next one is executed.

Selection - is the construct used to make decisions in a program. For example, if statements and switch/case statements.

Iteration - is the construct used to repeat sections of code.

↓

for loop, do until

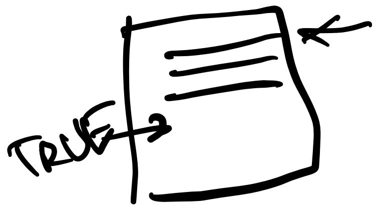
(c) Describe the difference between a for loop and a while loop.

for loop is count controlled
While loop is condition controlled loop.

We use for loop when we have a defined number of iteration.

[4]

File Handling



```
randomFile = open("random.txt")
while NOT randomFile.EndOfFile()
  data = randomFile.readLine()
  print(data)
endwhile
randomFile.close()
```

In the example above, the code prints out all of the data contained in the "random.txt" file.

- opens the text file "random.txt" using the identifier randomFile
- sets up a WHILE loop that runs until the end of the file is reached
- reads one line of data from randomFile into a variable called data
- prints out the contents of the variable data
- this loop repeats for each line in the text file, and prints out the contents of " random.txt" one by one
- once the end of the file is reached the loop stops
- randomFile is closed.

1D Arrays

A **one-dimensional array** allows a programmer to store multiple items of data under a single identifier. All items in an array are indexed meaning that the first item is at index position 0 and so on. Arrays are immutable meaning that once the size is declared, you can't then change it and add items on. Items in the array must also have the same data type.

array colours[5] -> creates an array of size 5
array colours = ["Blue", "Pink", "Green", "Yellow", "Red"]

0 1 2 3 4

colours[2] = "Purple" - this command will change the colour green to purple.
colours.length will return 5

To print out all the elements in the array we do:

```
                         5 - 1
for i = 0 to colours.length -  1
        print(colours[i]) → i
next i
```

The words below are stored in an array called `pets`.

rex	summer	fudge	mickey	rebel

.upper

Complete the algorithm to change the contents of the array to all uppercase.

```
01  i = 0
02  do
03      pets[i] = ......pet[i].upper.......................
04      i = i + 1
05  until i == ......5......................
```

[2]

do vs while

condition checked at the end

condition is checked at the start

The number of goals scored in each football match is held in an array called `goals`. An example of this array is shown.

```
goals = [0, 1, 3, 0, 4, 5, 2, 0, 2, 1]
```

Elliott wants to count how many matches end with 0 goals.

(c) Complete the following pseudocode for an algorithm to count up how many matches with 0 goals are stored in the array and then print out this value.

```
nogoalscount = 0
for count = 0 to (goals.length-1)
    if goals[count] == 0 then
        nogoalscount = nogoalscount + 1
    endif
next count
print(nogoalscount)
```

[3]

(d) The hotel has nine rooms that are numbered from room 0 to room 8.

name

The number of people currently staying in each room is stored in an array with the identifier `room`.

The index of `room` represents the room number.

Array `room`

Index	0	1	2	3	4	5	6	7	8
Data	2	1	3	2	1	0	0	4	1

The following program counts how many people are currently staying in the hotel.

```
for count = 1 to 8
    total = 0
    total = total + room[count]
next count
print(total)
```

When tested, the program is found to contain **two** logic errors.

Describe how the program can be refined to remove these logic errors.

① for count = 0 to 8

② total should be declared before the loop.

[2]

The text file 'mass.txt' on the right shows the masses of five objects. A scientist wants to write a program to analyse these values.

mass.txt

332.1
461.4
74.1
24.6
100.0

Write program code that reads each value from 'mass.txt' and puts them into a different element of an array called data.
Your program must contain:

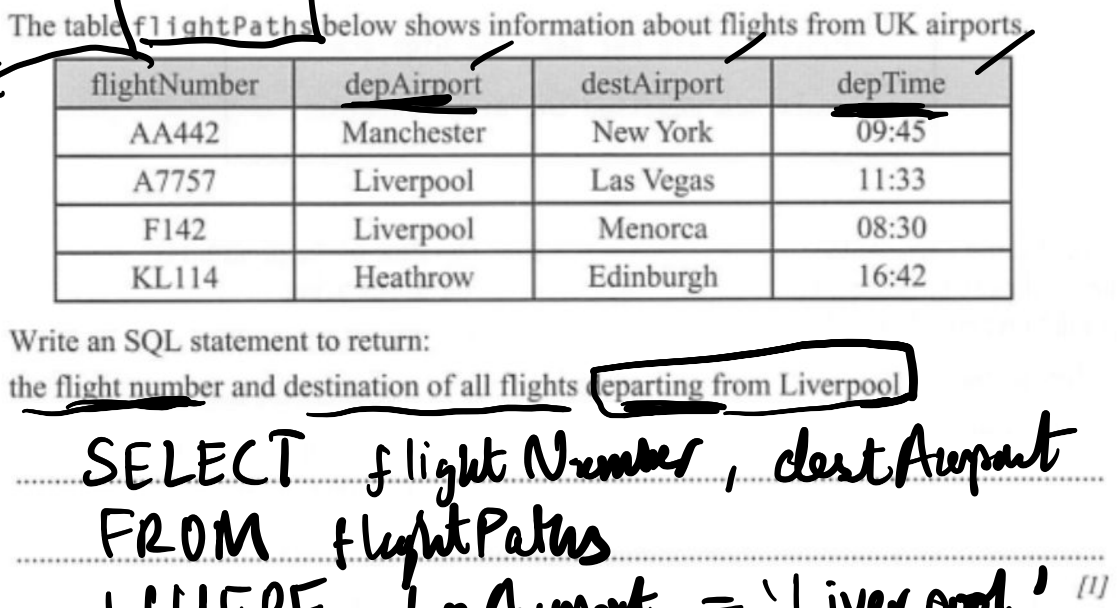
- a WHILE loop.
- the endOfFile() command.

```
i = 0
Values = open ('mass.txt')
while NOT values.endOfFile()
    data[i]= Values.readLine()
    i = i+1
endwhile
Values.close()
```

SQL

SQL (Structured Query Language) is a language used to access data stored in a database.

- SELECT identifies the fields to return from the database; (* means all fields)
- FROM identifies which table(s) the data will be returned from
- WHERE is an optional command that allows the programmer to include criteria, with only matching records being returned



The table `flightPaths` below shows information about flights from UK airports.

flightNumber	depAirport	destAirport	depTime
AA442	Manchester	New York	09:45
A7757	Liverpool	Las Vegas	11:33
F142	Liverpool	Menorca	08:30
KL114	Heathrow	Edinburgh	16:42

Write an SQL statement to return:

the flight number and destination of all flights departing from Liverpool

```
SELECT flightNumber, destAirport
FROM flightPaths
WHERE depAirport = 'Liverpool'
```

[1]

Function Vs Procedure

Procedures are a type of subprogram that do not return a value to the main program.
Functions are a type of subprogram that return a single value to the main program. This value then can be stored or used in the main program which is not possible with a procedure.

global variable.

parameters.

```
x = 20
procedure add(x, y)
  total = x + y
  print(total)
endprocedure

function addition(x, y)
  total = x + y
  return total
endfunction
```

local variable

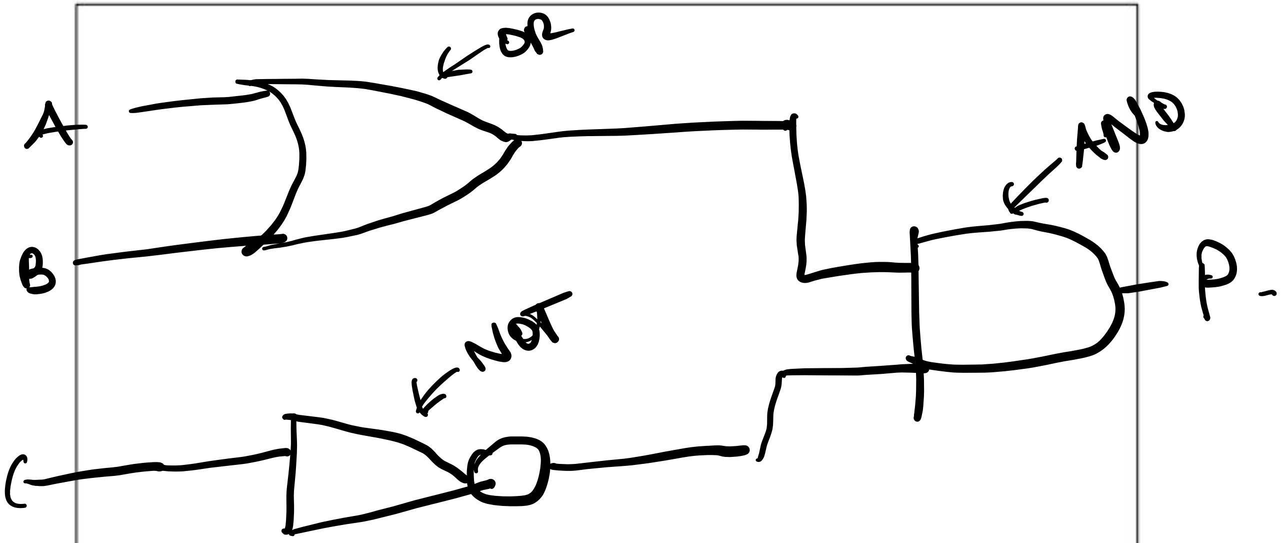
- The variables declared in a procedure or function are called **local variables** (e.g., the variable total). Local variables are defined in a subprogram and are accessible only in that subprogram in which they are defined in.
- **Global Variables** are defined at the start of the program and exist throughout the whole program and in all subprograms. These variables allow data to be shared between subprograms.

4. A fast-food restaurant offers half-price meals if the customer is a student or has a discount card. The offer is not valid on Tuesdays.
A computer system is used to identify whether the customer can have a half-price meal.
The table identifies the three inputs to the computer system:

Input	Value
A	Is a student
B	Has a discount card
C	The current day is Tuesday

(A or B) AND NOT C

(a) Draw the logic diagram for the computer system.



[3]

Defensive Design

Defensive design and testing help to make programs robust. Defensive design means thinking ahead about problems that could occur. It involves the following areas.

Anticipating misuse is thinking about ways that users could cause the program to fail. ✓
Authentication is checking the identity of a user. ✓
Input validation is checking whether data matches certain rules as it is input. ✓

Maintainability

Maintainability is ensuring that a program is as easy to understand and modify as possible for other future programmers. It involves the following:

- Use of subprograms - splitting programs down into multiple subprograms reduces the need to copy and paste code
- Variables and subprograms should use sensible naming conventions
- indenting is used to highlight the structure
- Commenting allows programmers to add notes to their program

The program below takes an input and returns a student's grade.

```
number = input("Please enter your mark")
total = 80
percentage = number/total * 100
if (percentage >= 70 AND percentage <= 100) then
print("You have achieved a grade A")
elseif (percentage >= 50 AND percentage < 70) then
print("You have achieved a grade B")
elseif (percentage >= 30 AND percentage < 50) then
print("You have achieved a grade C")
else
print("You have failed")
endif
```

Give two ways, using examples from the code, of how the maintainability of the program could be improved.

1. indentation

2. comment

[2]

Rajesh has written the procedure below to simulate 10 rolls of a 20-sided dice.

```
procedure diceRoll()
    for n = 1 to 10
        roll = random(1.0 to 20.0)
    next n
    print(roll)
endprocedure
```

There are two errors in his code.
Suggest how you would refine his procedure so that it works as intended.

- change random(1, 20)
- print(roll) should be inside the for loop.

[2]

The table contains four statements about programming languages.

Tick (✓) **one** box in each row to identify whether each statement describes a low-level programming language or a high-level programming language.

Statement	**Low-level**	**High-level**
The same language can be used on computers that use different hardware		✓
It allows the user to directly manipulate memory	✓	
It allows the user to write English-like words		✓
It always needs to be translated into object code or machine code		✓

Print

[4]

low

Compilers vs Interpreters

Translators convert high-level programming code into machine code so that the processor can execute it.

Compilers

- Translate every line of code in a program into machine code and run it afterwards.
- Produce an executable file
- Program can be run again without recompiling; simply run the executable file again
- Executable file can be distributed meaning that users will not see the source code
- Compiled code runs quickly

Interpreters

- Translate one line of code and run that line, repeating this process
- Do not produce an executable file
- Running the program again needs the interpreter to retranslate every line of code
- No executable file to distribute. Would need to share the source code to distribute the program.
- Interpreted code runs more slowly than compiled code.

IDEs

An integrated development environment (IDE) provides all of the tools that a programmer needs to write and test programs in one place.

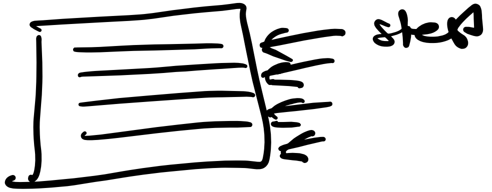
Editor

- A text editor to allow the programmer to enter or modify code in their chosen language.
- May include auto-suggestion of keywords.
- May include pretty printing to colour code keywords and automatically indent code

print

Error diagnostics

- Tools to allow the programmer to find and fix errors.
- Breakpoints stop the program at a specific point.
- Stepping allows the programmer to run the code from this point one line at a time.
- Variable contents can be checked.



Runtime Environment

- Allows the programmer to run the code from within the IDE.
- The program output can be seen without opening additional programs.
- ~~May involve the use of a virtual machine.~~

Translators

- Converts the high-level code into machine code to allow execution by the processor. ✓
- IDEs include interpreters or compilers (or both).

switch case

The vending machine can be in one of three states: on, off or suspended. A user can change the state of the vending machine by using the following algorithm.

```
newstate = input("Enter the new state : ")

switch newstate:

    case "on":

        statevalue = 1

    case "off":

        statevalue = 2

    case "suspended":

        statevalue = 3

    default:

        print("Invalid state")

endswitch
```

Rewrite the algorithm to perform the same actions using IF statements in place of the switch statement.

$\boxed{\text{Pseudocode !}}$

```
if newstate == 'on' then
    statevalue = 1
elseif newstate == 'off' then
    statevalue = 2
elseif newstate == 'suspended' then
    statevalue = 3
else
    print('Invalid state')
```