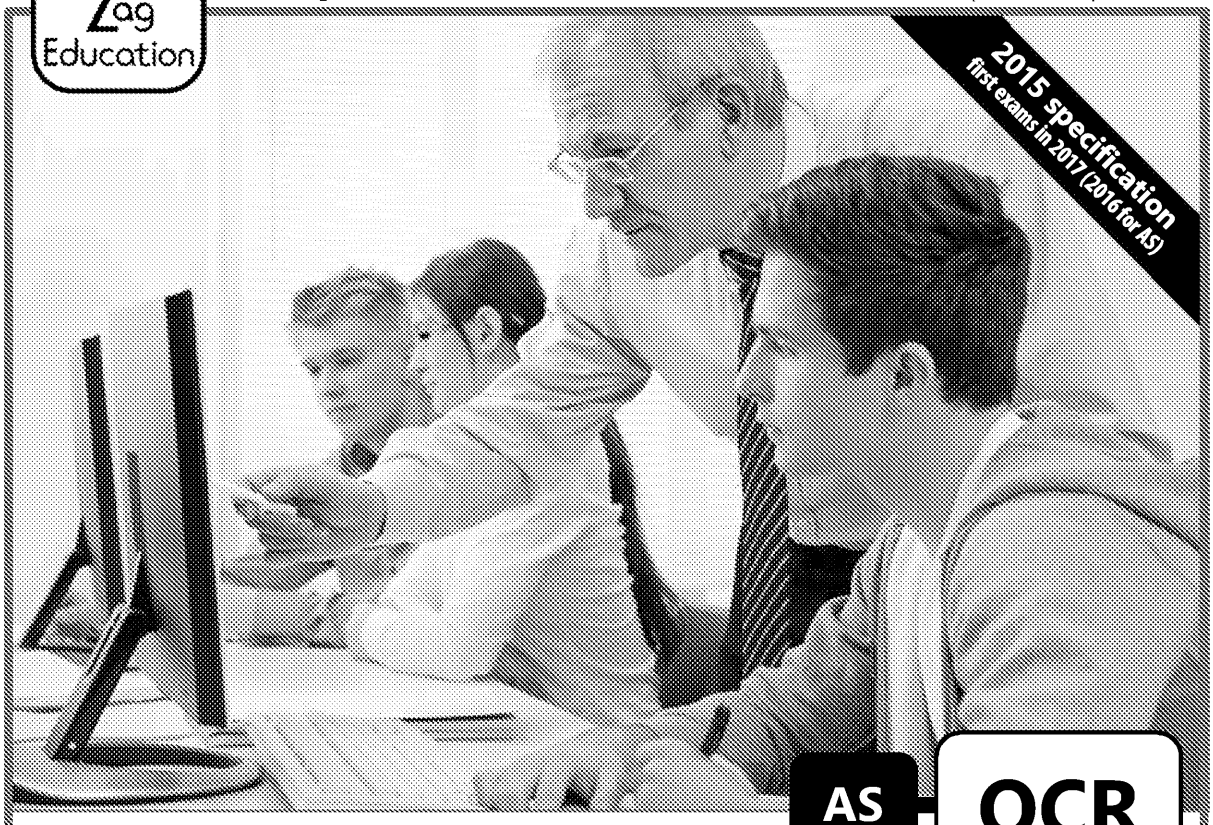
Zig Zag Education

Computer Science

AS | OCR | H046



2015 specification
first exams in 2017 (2016 for AS)

AS **OCR**

Revision Guide

for AS OCR Computer Science

Component 02

zigzageducation.co.uk

POD 6282

Contents

Teacher's Introduction

This revision guide has been written to support the OCR AS Computer Science specification (first teaching from September 2015, first exams in June 2016).

It summarises the essential theory required for the AS Component 02, in which students are assessed with a 1 hour 15 minute written examination (70 marks).

Component 02 and this resource covers the following topics from the OCR AS specification:

2.1 Elements of computational thinking
2.2 Problem-solving and programming
2.3 Algorithms

Note that an equivalent resource is also available for the AS OCR Component 01 examination.

Remember!

Always check the exam board website for new information, including changes to the specification and sample assessment material.

Each section includes student notes, examples, diagrams and examination-style questions. Example answers to all of these questions can be found at the back of the resource. *Note that credit should also be given for any valid responses that are not explicitly included in this resource.* There is also a revision progress grid which students may find useful in the lead up to their exams.

Programming concepts are exemplified throughout using pseudocode and a number of high-level programming languages including Java, C++, Visual Basic and Pascal.

March 2016

REVISION PROGRESS TRACKER: COMP

Specification Topic	Confidence Level (1-5)				
	Date:	Date:	Date:	Date:	
2.1 – Elements of computational thinking					
Thinking abstractly					
Thinking ahead					
Thinking procedurally					
Thinking logically					
2.2 – Problem solving and programming					
Programming constructs					
Global and local variables					
Modularity, functions and procedures					
Use of IDE					
Waterfall lifecycle					
Agile methodology / extreme programming					
Spiral model and rapid access development					
Black and white box testing					
Alpha and beta testing					
2.3 – Algorithms					
Bubble sort					
Insertion sort					
Binary search					
Linear search					
Stacks					
Queues					

2.1 ELEMENTS OF COMPUTATIONAL

2.1.1 THINKING ABSTRACTLY

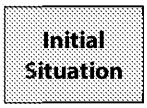
Computational thinking

Computational thinking is a problem-solving process that creates a generalised solution decomposition, abstraction, modelling and the use of algorithms. Computational thinking

- Problems are formulated so that a computer can be used to solve them
- Logically analysing and classifying data
- Representing data models using abstraction techniques
- Algorithms are used to automate solutions

Problem-solving

Problem in reaching a desired outcome from an initial situation. The first stage in problem-solving is to gain a good understanding of the problem that is to be solved.



The next stage is to create a definition of the problem, which involves the following components:

- The initial situation
- The desired outcome
- The resources available to solve the problem
- The ownership of the problem (who will be involved in planning the solution an

Planning a solution involves deciding upon a plan of action or strategy to solve the proble solve the problem on paper using a range of assumptions to simplify the problem.

Most problems can also be broken down into a series of smaller problems or sub-problem known as decomposition.

Decomposition

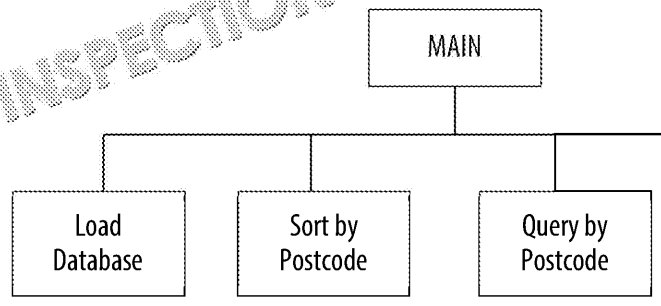
Example: Addressing labels

A travel agent wishes to send a brochure to each of their customers in a certain area, based on their postcode.

Solution: Load the database and create a query for the customer file based on a selected postcode; then print out the envelope with the relevant address. The hierarchy chart below shows the initial decomposition of the problem.

Decomp approac breakin steps, w further

The sma understa problem



Each of the steps can then be broken down further to add more detail for the solution; s
functions include additional detail:

- Switch on printer
- Load envelopes into printer tray
- Run postcode query on database
- Select print query from database

The other functions: 'load database', 'sort by postcode' and 'query by postcode' can also

The final design should include enough steps for the designer to create the algorithm fo
steps need to be added, hence the term stepwise refinement.

Decomposition advantages listed below:

- Breaking the problem into parts helps to clarif[illegible] hat needs to be achie
- Each stage of the refinement proce[illegible] ea[illegible] maller sub-problems that are ea
- Some functions or parts of [illegible]olution might be reusable
- Breaking desig[illegible] allows more than one person to work on the solutio

Algo

An algo[illegible]m can be expressed using pseudocode; this is a written list of steps that are required to complete a process, not connected to any particular programming language.

An **algorithm** is a problem. It can be independent of a

Once a problem has been specified a series of steps or an algorithm can be created to produce a result.

Computer-science-type problems are solved using algorithms as shown in the example on the right.

Algorithms can be used to describe non-computer type problems as shown in the travel example below.

Problem – Need to travel to Paris

1. Contact travel agent
2. Choose to travel by airplane
3. Purchase ticket
4. Collect travel documents
5. Pack luggage
6. Travel to airport
7. Board correct plane
8. Collect luggage
9. Use taxi to get from airport

Algorithm to calculate
the keyboard.

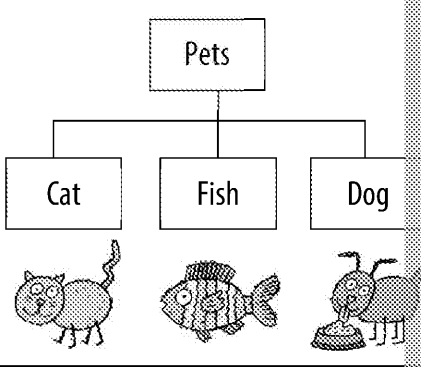
```
START
    # Data decl
    REAL
    INTEGER

    Total ← 0
      # Loop to
      FOR X =
       OUTPUT
       N ← USE
       Total ←
      END

      Average
      OUTPUT "A

END
```

The nature of abstraction

In computer science abstraction is an process of including [illegible] important feature[illegible] so[illegible] problem; this reduce[illegible] mplexity of the system by removing unnecessary details.

The benefits of using abstraction techniques are that it is easier for the programmer or user to view, to modify and to maintain the solution as they are not distracted by excessive detail which is hidden from them.

The hierarchy diagram below is an abstract generalisation for pets



Abstraction and reality

Abstraction is one of the key features within object-oriented programming, where the programmer reduces complexity by hiding all but the essential data about an object.

An object, which is based on a class, is a model of a concept, process or real-world item th is relevant to the application.

Variables are used in programs to perform calculations and can represent real-world value as well as intermediate values in a calculation.

The abstraction method is widely used in object-oriented programming languages; for ex called 'CalculateTax' in their 'Worker' class, which uses WorkerID as an input parameter ar value. The programmer does not need to know how the calcul s carried out; they sim the method and the input parameter and the format of th put.

Note that for function abstraction th computational method used is hidden, unlike in or library functions is an e of functional abstraction; the user simply calls the fu knowledge the code within the function.

For exan ng a built-in square root function:

SQRT(16) will return the value 4, so x = SQRT(16) will bec

Devising abstract models

Abstraction is a key feature in computer science where abstract models are created for re essential that sufficient detail is included from the abstraction process to create a model t required level of accuracy. This automation process is based on the following:

- Creation of algorithms – which includes the breaking up of the problem into sub needed to solve each sub-problem.
- Implementing the algorithms in program code – which includes conversion betw instructions of the programming language chosen.
- Implementing the models in data structures – chosen data structures should be used data structures include arrays, files and record/ strcuts.
- Executing the code – once the code has been created it should be executed to e to ensure it operates as expected.

Abstraction layers

Computer systems make use of the concept of layers (or levels) of abstraction to represen layers. The functionality of the various layers is hidden from the other layers.

The diagram below shows a view of the abstraction layers for a puter architecture.

OS and Applications
Kernel
Assembler
Firmware
Hardware

2.1.2 THINKING AHEAD

It is important to think ahead when considering any task that needs to be carried out; this app
with computer solutions.

Identify inputs and outputs

It is important to be aware of the input and outputs needed when planning a computer system.

The inputs and outputs will depend on the application where a computer system widely modelled is an online sales operation.

Example

An in-ca system can be used as an example of the inputs and outputs needed by a computer system.

A touchscreen is used in the screenshots shown on the right, where:

- inputs are added by pressing options on touchscreen menus
- outputs are displayed on the screen

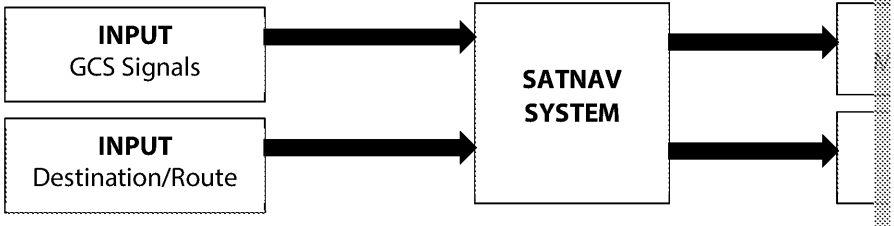
The inputs entered are the destination postcode and the calculation criteria from which the system will determine a suitable route, and give instructions from the start point.

The current position input signal is read from a GPS signal.

1 – input the

2 – select fa
calculati

The block diagram below shows the commonly used input and outputs for a satnav syste



Preconditions

It is necessary to determine the precondi ssary for devising a solution to a problem.

Precondit re as variables or parameters within an algorithm and so t e initialised to the correct values to ensure that the solution operates as expected.

The example on the right initialises two variables prior to using them in a while loop.

```
#Init
  F
  cou

WHILE
  F
  co
ENDWH
```

Caching

Cache memory is normally fitted to the motherboard and is the highest speed memo
caching process employs the *'thinking ahead'* concept, where:

- it caches (or stores) data and instructions that are likely to be needed agai
- this data is be stored in cache memory rather than RAM where it can be ac
- therefore, the rate at which programs are executed is speeded up

A typical use for cache is on the Internet, where the first time a website is loaded its
temporary files in cache memory. This can be useful as when the site is visited again
cache memory rather than by downloading from the Internet.

Cache benefits	
Cache is faster than main memory	Cache memory is sm
It stores regularly used instr[illegible] data so they can be processed fast[illegible]	Cache memory is ver

Re[illegible] components

Reusable library software

A programming task can be created faster by including reusable software; the code i
The library code has been thoroughly tested and optimised.

A program library is a series of predefined and compiled routines that are available f
method for reusing fully working and tested code. The use of library code can save t
easier to read and understand; typical uses include:

- Conversion of data into well-known formats
- Standard scientific functions
- Accessing external storage
- File management
- Input/output Interface

An example of library software is the C Library (stdio.h) which is used to perform inp
physical devices such as keyboards and printers.

In **C++** the input / output library is called using the command: #

2.1.3 THINKING PROCEDURALLY

Components of a problem

The components of a proble[illegible] b[illegible]entified using abstraction, where the det
removed or reduc[illegible] th[illegible]roblem is represented in a way that is straightforv

[illegible]he [illegible]ssary details in the problem are removed, then the problem and
[illegible]sed. Therefore, it is possible that the problem or some of the components
organisation before.

Components of a solution

Composition is the opposite of decomposition; it is the process of creating a system by identified in the decomposition abstraction. This process involves:

- Writing sub-procedures for each task identified in the decomposition;
- Linking these sub-procedures to create compound procedures;
- Creating the necessary data structures to support the compound procedures

Order of steps in solution

The order of steps taken in a solution can be crucial. In an obvious example, the progr has been read into the system.

To ensure the solution is carried out in the correct order it can be traced through by ha where a program is tested by hand and variables are recorded as and when they chang contains columns for the calculation result for each of the variables. Inspection of the the code can then be carried out.

Use of sub-procedures

Procedural abstraction is based on programming where large programs are written by procedures; this approach applies to any programming language, although the units a methods in Java and functions in C and many other languages.

The procedure is a named block of code, where the actual data and values used in the abstracted. The use of local variables declared within a procedure helps to ensure tha 'black box' that can be used by the operator without an understanding of its process.

The advantages of using sub-procedures are:

- The use of sub-procedures aids readability making main procedure appear le
- It is possible that sub-procedures can be reused to solve other problems
- A sub-procedure can be treated as an object that has its own task
- Development of a solution can be improved as the code of each sub-procedu stand-alone task

2.1.4 THINKING LOGICALLY

Logical decisions and program flow

In many programs there will be a need to make decisions, such as a selection or iteration construct.

The simple computer-controlled greenhouse temperature flowchart has been included to demonstrate logical decisions, as explained below:

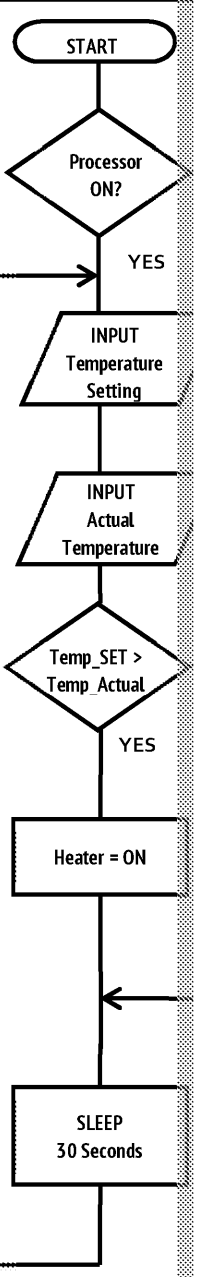
1. Decision point 1 – at the start of the routine system checks if the processor is running, if not the program will end.
2. The control temperature setting is input into the system; it will be in the range 17 to 27 °C.
3. The actual temperature from the sensor is then input automatically into the system.
4. Decision point 2

   If the control temperature setting is higher than the actual temperature the heater is switched ON to raise the temperature of the greenhouse.

   Else the heater is switched OFF in which case the greenhouse will cool down.
5. There is then a pause (or sleep) for 30 seconds to allow the system to respond to the heater input, before the process is repeated.

Note this is a very basic system to simply demonstrate some logic in a non-trivial program; more advanced systems might include open/close window motors or computer-controlled fans.

The Boolean expressions used to control program flow will output true or false.



2.1 – Progress Check

1. Define the following terms: (4 marks)
   (a) Computational thinking (b) Decomposition
   (c) Algorithm (d) Abstraction
2. List the advantages of decomposition (4 marks)
3. Describe two benefits and two drawbacks in using cache memory (4 marks)
4. Describe the term 'program library' and the benefits of using it (4 marks)
5. Describe the advantages of using sub-procedures (4 marks)

2.2 PROBLEM-SOLVING AND PROGR

Note: Specification points to the benefits of using a procedural/imperative language f

Procedural/imperative languages – High-level languages include imperative languag statements or instructions are executed in a sequence or order as defined by the prog known as procedural languages, make use of subroutines and functions to aid readabi

Low-level languages are imperative as all instructions are executed in a set sequence level is translated into numerous machine code statements prior to the execution of t

2.2.1 PROGRAMMING TECHNIQUES

Programming Basics

The s to simple problems can be written in pseudocode using one or more of t and iterations.

Sequence – In a sequence structure the program executes each function or statement in order.	**# Sequencing Exampl**

The majority of code in a program is executed sequentially.	`Input Length, Inp`
	`Area ← Length *`
	`Output "Rectangle`

Selection – A selection structure is where the program executes different actions result of a comparison.

IF-THEN statements are used to execute one block of code when a Boolean condition is TRUE, there is no alternative branching when the Boolean condition is FALSE.	**# IF-THEN Example**
	`IF (X > MAX) THEN`
	`X ← MAX`
	`END IF`

IF-THEN-ELSE statements are used to execute one block of code when a Boolean condition is TRUE and an alternative when the Boolean condition is FALSE.	**# IF-THEN-ELSE Exam**
	`IF (Age >= 18) TH`
	`OUTPUT "Yo`
	`ELSE`
	`OUTPUT "No`
	`END IF`

CASE statements (termed switch statements in progra such as JAVA/C++) are a variation of an IF-THEN-ELSE statement where several IFs are required. Note: Python does not have an actual Case statement but the same result can be achieved in Python using the 'elif' stat	**# CASE or SWITCH Ex**
	`CASE Weekdays`
	`1: THEN Day`
	`2: THEN Day`
	`3: THEN Day`
	`4: THEN Day`
	`5: THEN Day`
	`END CASE`

Iteration

ⓘ **Iteration –** Iteration is where a program executes a statement or statements that
condition is satisfied.

FOR LOOP
This iterative control method is useful when the same instructions or calculations have to be carried out for a known number of iterations.
	#Example FOR LOOP
	`Total ← 0`
	`FOR X = 1 TO 10`
	`    Total ← Tota`
	`ENDFOR`

WHILE LOOP
In this iterative control method technique, the loop operates when a condition is satisfied at the start of the loop and stops when this is no longer true.
	#Example WHILE-LOO
	`F ← 1`
	`counter ← 10`
	`    WHILE count`
	`    F ← F + 1`
	`    counter ←`
	`ENDWHILE`

REPEAT UNTIL
In this iterative control method technique, the loop operates at least once; then the condition is tested at the end of loop and repeats until the condition is no longer true.
	#Example REPEAT-UN
	`REPEAT`
	`    OUTPUT "Ente`
	`    N ← INPUT`
	`    OUTPUT N`
	`UNTIL N = 11`

Local and global variables

ⓘ **Variables** are used in computer programming to store data values of a particular d
some typical data declarations for some widely used programming languages.

C#	Java	Python

`bool blogic;`
`int intVal;`
`char chVal;`
`double Sum;`
`string stVal;`
	`boolean blogic;`
	`int intVal;`
	`char chVal;`
	`float Sum;`
	`String stVal;`
		Variables are declared
		assigned in Python, so
		Distance = 10.5
		will be declared as **flo**
		point.

A variable can be declared as either local or global.

Local variables are only visible or in **scope** within the function in which they are decla
accessed from everywhere within the program.

Local variables in subroutines Where local variables are declared and used in a subroutine, they are only in existence while the subroutine is being executed and are only accessible or in scope within that subroutine. It is good practice to use local variables rather than global variables in subroutines and functions as it easier to trace the content of a local variable in a large program and the subroutine retains modularity where only parameters and not global variables are passed to it for execution. Local variables are more efficient than global variables as you free up memory each time you finish using the local variable.	**Global variables in a p** Where global variables program they can be acc program in contract to l in the program block in Global variables shoul throughout the compl some programming la unless declared in a fu Note this is not the ca local unless declared e

2.2 – Progress Check

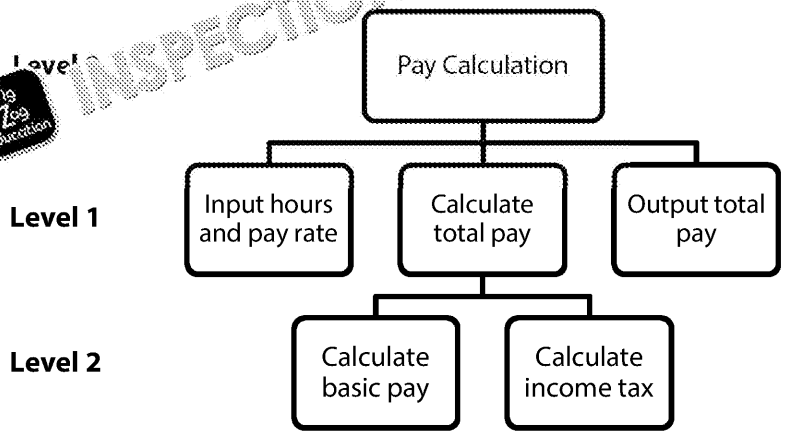
1. Explain why it is good practice to use local variables rather than global variab

Modularity, functions and procedures

Modularity and structured programming

Structured programming is a type of procedural-oriented programming where the pro help reduce development time and ensure that the program is easier to understand an

Hierarchy charts are used to show the details of the modular structure in the program shows the modules involved in designing a simple pay calculation.



Advantages of modularity

The structured approach to programming reduces the complexity of the task and has t

1. Breaking down large programming tasks into manageable subtasks means th several programmers, which saves development time.
2. Program test and debug time is reduced, where modularity helps to reduce t correcting the errors that may occur.
3. Programs are easier to understand and, therefore, easier to maintain; also a introduction of an additional module.

Procedures and functions

Procedures and functions are useful in helping to provide a structure to create logical
There is a similarity between procedures and functions, in that they both support the
calling the function or procedure many times, rather than continually writing out or co

This approach reduces the amount of code and creates a more readable solution; they
statement blocks that are executed by using an identifier in one or more places throu
code is linked to the runtime version of the code at compile time.

The difference between functions and procedures is language-dependent, but general
procedures do not.

Some programming languages, such as C, only use fu in this case some functio
operate like procedures and the return typ is vo

Advantages of using Pro es a unctions

1. One o advantage of using procedures and functions is that less co
   el ment time will be reduced.
2. Also, having less code makes the solution easier to read and understand whe
   the program.

Built-in functions

Built-in functions are used in computer programs to create efficient code to solve con
functions have been provided by the computer program developers.

Software documentation gives a list of the built-in functions available with guidelines

Typical example is the square root function **sqrt(val)** available in Pascal / Java / C# / V

So sqrt(9) returns the square root of 9 which is 3

Programming languages also allow the user the option to create their own functions v
square function:

```
{Define a Function to square a number}
FUNCTION Square (Val)
   S ← Val * Val;
   Return S
END FUNCTION
```

Parameters passing by value and reference

A **parameter** is a variable that is used as a data input or an argument that can be
be passed by value or by reference.

Parameter passed by reference	**Parameter passed by**
In this case the parameter passed to the function is a memory reference. If the value of the variable is changed in the function then value at the memory reference will also change.	In this case the param of the original data. If the value of the vari the new value is not w

Visual Basic example by **reference**
`Sub SubExample(ByRef MyParameter)`
`MyParameter = 8`
`End Sub`
`Dim Ref`
MyP is an alias for MyRef
The subroutine assigns MyParameter = 8
So MyRef is changed to 8 when subroutine called
	Visual Basic example
	`Sub SubExampl`
	`MyParame`
	`End Sub`
	`Dim Ref`
	`MyRef = 555`
	MyParameter is now a
	The subroutine assign
	But MyRef is unchange

Returning a value from a function

An example of a simple subroutine (VAT) is shown below; when the function is called
based on the input parameter.

VAT example in C programming language	**Function defined**

The function is named VAT and contains one input parameter named 'purchase_price'.
The parameter is data type double (floating point) and the data type returned is also double.
The function calculation details are within the {} brackets
	`double VAT(double p`
	`{`
	`return 0.2 * pu`
	`// cal`
	`}`

Once the function has been defined it can be called as shown on the right.	`total_price = purch` `VAT(purchase_price)`

Therefore, if the purchase_price is 100 then **VAT** function returns 20.	`purchase_price =`
	`total_price =`
	120

2.2 – Progress Check

2. Describe the ... using procedures and functions (2 marks)

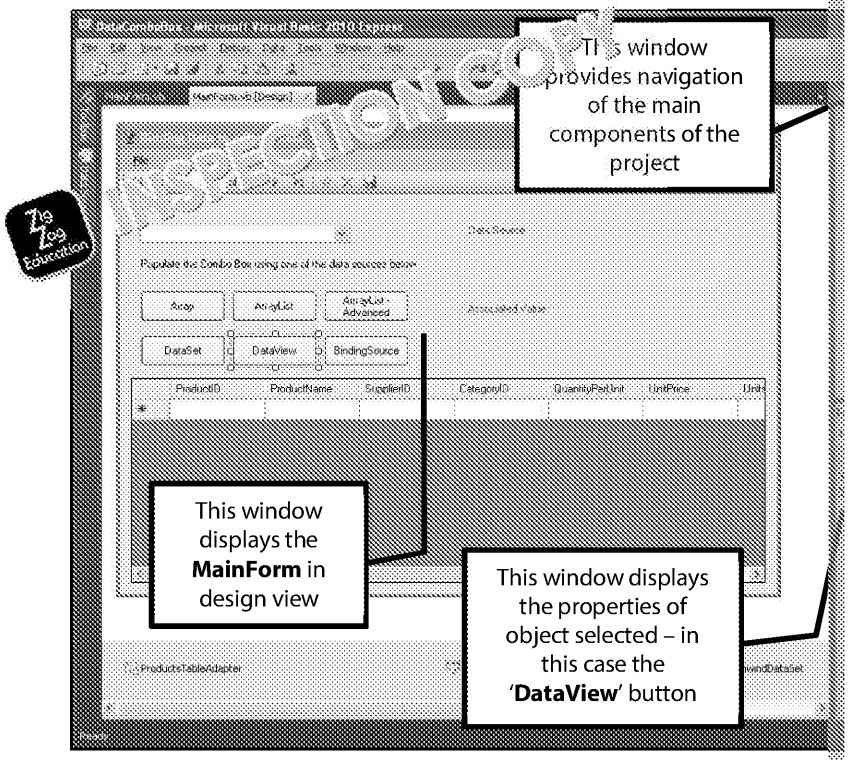
... difference between passing a parameter to a function by referenc

IDE for development and debug

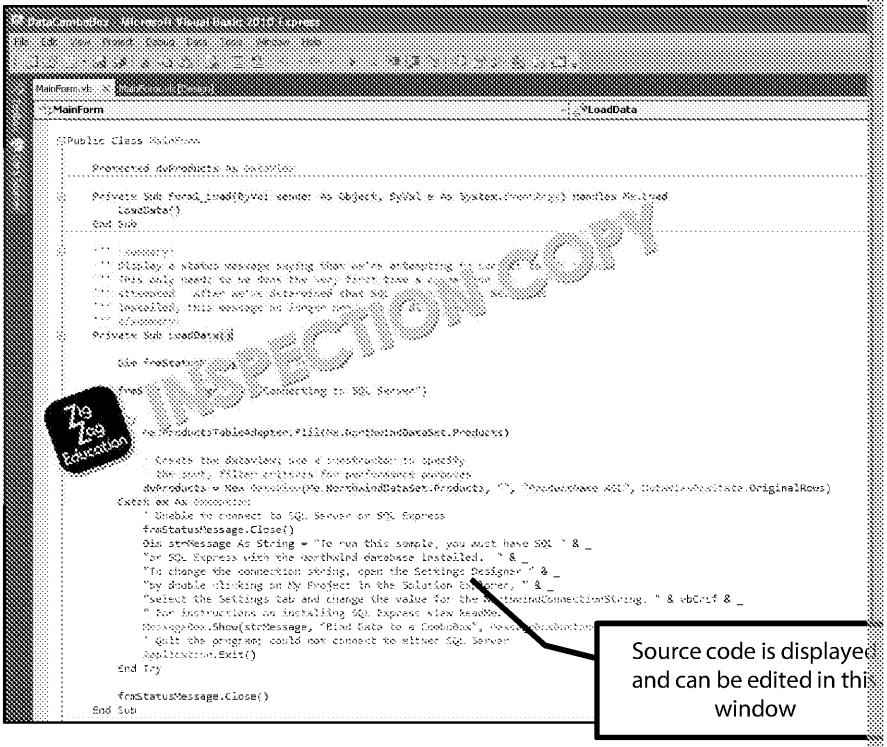
Integrated development environments (IDE) are widely used to develop software and contain a range of features, such as: source code editor, debug and error diagnostics, run-time environments, translation tools and automatic documentation facilities.

Microsoft has created a range of Visual Studio IDE software to support their Visual Basic, C++ and Java software.

In the screenshot below there is a sample of a Main Form in Visual Basic for the Micro
design view. The main menu contains the options to manage the project and the run-t



Source Editor – used to create or modify the source code for a computer program.



Error diagnostics and program development

As described earlier **translators** are used within the IDE system to compile or interpret the source code.

The IDE will also provide a series of debug tools that are used during the translation process to find possible errors.

Debug tools or a debugger is a software program that is used to find 'bugs' or errors in other programs. There are various options that can be chosen when debugging a program as shown in the screenshot on the right.

These debug tools or utilities are provided to support the programming language that is being used by allowing the developer to operate the code in a run-time environment.

Typ[illegible]g [illegible]ls include:

- **Single step** operation by the debugger works through the program code one [illegible] view related data before and after execution to help them to locate problem
- **Watch** – variables can be monitored to see what values they are changing to both **run-time and logical errors** where a variable might occasionally be ass
- **Breakpoints** can be added to programs by the debugger; they simply stop th allows the developer to gain knowledge of how the program is operating an to locate errors.

Auto documentation is a feature available in some integrated development environm the software project that has been developed. The documentation provided is depend language used, but would normally include a text file presentation of details such as v subroutines, macros and programmer comments.

One of the main advantages of using automatically created documentation is that it w the latest code created.

2.2 – Progress Check

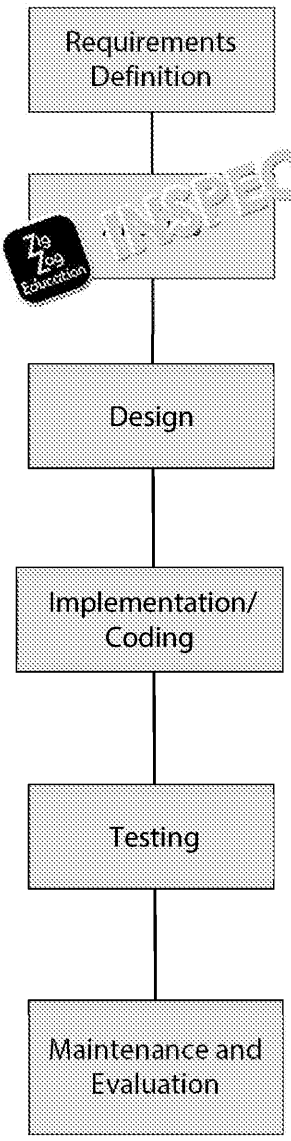
4. List the features contained in an integrated development environment (5 mar
5. Explain how breakpoints can be used to debug a program (3 marks)

2.2.2 SOFTWARE DEVELOPMENT

Understand alternative methodologies and their merits and drawbacks

Software development

Software development – software needs to be developed in stages in a structured are well managed and meet the system requirements for the organisation; the sta



Requirements Definition – after deciding to de organisation will deter e the aims and object

Analysis – this phase follows on from feasibility **requirements**. Further information is then obta questionnaires, observation and inspection of d **criteria** are then listed for use in the evaluation

Design – during the analysis phase the file stru needed by the system were defined, so in the de needs to be created.

This will normally include: diagrams of the syste description of processing, design and format of s hardware and software platform required and a

Implementation/Coding – this phase is where a produced. The stages of implementation are dep be produced.

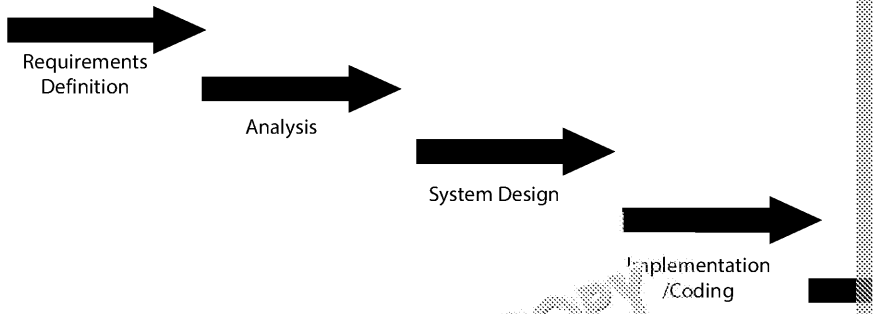
In the case of software development, the code g design has been completed. Code generation is been completed to a good standard. Software m units, which are **unit tested** by the developer to

Testing – the test plan created in the design phas operation and no unexpected results. Next stage i tested in a stand-alone way to be **integrated** toge

Maintenance and Evaluation – once the system criteria will be used to evaluate or measure syst occur will be fixed using **corrective maintenanc perfective maintenance** and new requirements **maintenance**

Waterfall model

Waterfall model – one of the earliest methods of structured systems developmen



Uses logical approach with little overlap and 'splash-back' acceptable between phases

The emphasis using the waterfall method is on planning, target dates, budgets and th at one time. This is a formal method with extensive written documentation, and appro technology management before beginning the next phase.

This approach is most suitable for the development of mainframe-based or transactio are clear objectives and no pressure for an immediate implementation. Known as the completed, you progress onto the next phase with no turning back.

Waterfall methodology advantages	Waterfall m
Ideal for supporting inexperienced project teams and where requirements are defined	Dependent upon clear little 'splash-back'
Orderly sequence of development ensures quality documentation with reliability and maintainability of the developed software	Produces excessive do as the project progress
Progress of system development is easily measurable	Missing system compo design and coding
Project progresses forward, with only slight movement backward	System performance c almost fully coded

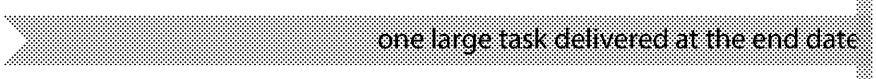
Agile methodology

Agile methodologies are a range of alternative methods to traditional software d an iterative and incremental method to help software developers respond to chan

The agile process is to break down the tasks into functional elements and deliver them duration; the diagram below shows a project broken down incrementally.

Each increment consists of various phases such as analysis, design, coding and testing can be involved in changing requirements.

The more traditional approach is to develop the software as one unit and deliver it all



The user gets the completed system at the end date, when it is too late to modify the

Extreme programming

Extreme programming is an agile software development methodology that conce
correct implementation. As an agile method, there are frequent iterations in the p
and improved during each iteration.

The main features of the extreme programming methodology are:

- **Pair programming** – with this technique two programmers work together a
writes the code while the other programmer offers advice on how to improv
functionality
- **Unit testing** – all code is tested continuously; white box testing is possible
programmers who have a detailed understanding of the solution.
- **Code reviews** – carried out on a regular and continuous basis, in some
programmers on a line-by-line basis to ensure high-quality coding.
- **Client communication** – important feature that ensures that changing requ
programmers are fully aware of any modifications that might be required.

The advantage of using extreme programming is that it concentrates on creating well
also it is created quickly and modules become available for use by the client as they a

Disadvantages are that the client has to ensure that they are represented on the deve
code and to discuss any potential changes; also the emphasis on coding rather than d
documentation, making it an unsuitable methodology for larger projects.

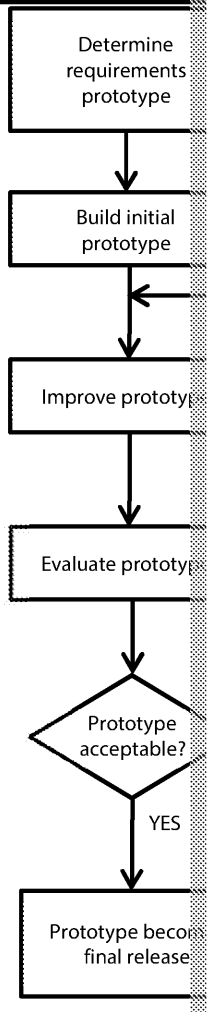
Rapid application development

The rapid Application Development (RAD) methodology was developed in response to the need to deliver systems very fast.

RAD uses a range of management procedures that support a speedy development, including:

- **Prototyping** is an iterative approach based on producing a working model as fast as possible. The prototype is then refined using feedback from the software developers as well as the subsequent users of the system.
- **Incremental** development based on refinement is the hallmark of this methodology, in which prototyping and iteration go hand in hand; in this case the throwaway prototype approach is used to create production software.
- **Time-boxing** is a time management technique that is focused on completing each task within a fixed time span. Consequently the development team will do their best to get a solution within that time frame, although in some cases this will be at the expense of functionality

The flowchart on the right shows the RAD prototyping process.

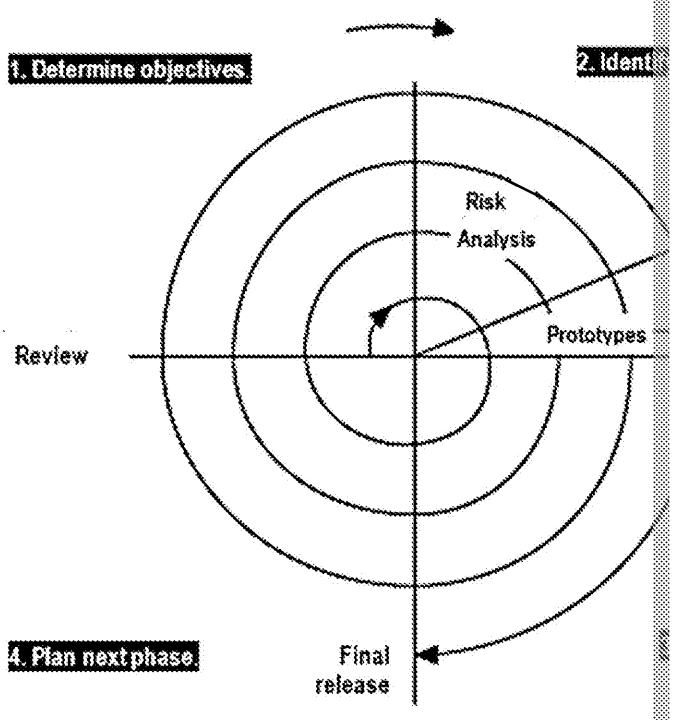


RAD Strengths	RAD
The production version of an application is available earlier than using other methodologies such as the waterfall approach and, therefore, tends to cost less to produce Concentration on the essential elements needed for the user with an emphasis on fast completion	The emphasis on speed overall system quality Potential for inconsister to detail in respect of ac

The RAD methodology is not suitable for safety-critical systems; but is an appropriate the system are not well defined and the development team is authorised to make des detailed consultation with their senior managers.

Spiral model

The spiral life cycle model is a combination of the **waterfall** model and **prototypi**



Each cycle of the spiral creates a prototype; this prototype is then evaluated and the p acceptable solution for the client.

This method is often used where it is difficult to define project requirements or where the risks are high. The following stages are carried out:

- **User/Client Communication** – frequent discussion with developer to establi
- **Planning** – is where timelines and project alternatives are considered
- **Risk Analysis** – is based on factors such as cost of development, technical ri customer might decide to abort a project that is likely to be unsuccessful
- **Coding and Engineering** – this task is to build a representation of the solutio
- **Creation and Release** – the solution is created, tested and installed for use support and full documentation will be provided for the final solution.
- **User/Client Evaluation** – is where client feedback is obtained based on the implemented; the spiral is repeated until the client decides they have an acc

Spiral Model Advantages	Spiral M
Large amount of risk analysis ensures that issues are addressed early in project development A software prototype is created early in the life cycle and refined with each spiral iteration	Highly skilled develop risk analysis Development costs car prototypes being creat collaboration

The spiral method is normally used for mission-critical projects, as well as projects wh game developments.

Following and writing algorithms

An algorithm can be expressed using **pseudocode**, this is a written list of steps that ar connected to any particular programming language.

Note that pseudocode can be written in many styles, but it should be in sufficient deta code can be written. In the examples below, comments are made using the **'#'** Symbol; fo goes after the

The solutions to simple problems can be written in pseudocode using one or more of t assignment, selection, and iterations, as shown in the following examples:

Example 1: Algorithm to calculate the area of a square from a value entered by the us

```
START

        OUTPUT "enter a number"
        Length ← USERINPUT
        Area ← Length * Length
        OUTPUT "Square = "; Area

END
```

Example 2: Algorithm to calculate the area of all the squares with values 2, 3, 4 and 5

```
START
      FOR Length = 2 TO 5
          Area ← Length * Length
          OUTPUT "Square = "; Area
      ENDFOR
END
```

Example 3: Algorithm to calculate the average of 10 numbers that are input from the

```
START
 Total ← 0
 Count ← 10

      # While loop takes inputs from keyboard
      WHILE Count > 0
          OUTPUT "enter a number"
          N ← USERINPUT
      Total ← Total + N
      Count ← Count -1
      ENDWHILE

      Average ← Total / 10
      OUTPUT "Average = "; Average
END
```

Test strategies

The test process should take place throughout the coding process using **black or white box testing** to ensure that all modules operate as expected.

Black Box Testing

This method of testing can be carried out without a thorough knowledge of the internal workings of the item being tested.

For example, when black box testing is applied to software engineering, the tester would only need to know what the normal inputs and the expected outputs should be, but not how the program arrives at those results.

White Box Testing

This method is based on the tester having th knowledge and programming skills to identif all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs, the aim being to:

- [illegible] each line of code in the program
- pass through every branch statemen
- cover all possible logical statements

When the coding has been completed and is error-free it is time to move on to **alp and beta testing**:

Alpha Testing

Alpha testing reveals errors and omissions that are made during development and is normally performed by the programmers that produced the software.

The test plan is used as the basis for testing to make sure the system operates to the specification; any unexpected results are corrected.

Beta Testing

Beta testing involves releasing the fully tes software to a limited audience to test in a working environment, where:

- using real live data may throw up unexpected faults
- integrating with other software applications may indicate interface errors

Most common errors should get shown up a can be corrected before final product releas

Test programs

The test plan created in the design phase is used to ensure correct operation of the so Next stage is for the software modules that have been tested in a stand-alone way to complete system.

The results of the testing are printed out and delivered to the customer, to show that Testing should be documented so that there is some evide[illegible]at it worked correctly to the software the test plan can be used again t[illegible]ensure that everything is still worki

When testing, it is important to [illegible] test data that works for erroneous as well as no

- **Normal Da[illegible]** data values to represent normal inputs
- **tre[illegible]ta / Boundary Data** – acceptable input data on the extreme limi
- **[illegible]oneous Data** – data outside of normal input range or in a different forma

Test plan example – to ensure product price in the range of			
Test	**Test Title**	**Test Data**	**Expected Results**
1	Product Price Entry Normal Data	£29.99	Normal input accepted
2	Product Price Entry Extreme Data	£10.00	Extreme input accepted
3	Product Price Entry Extreme Data	£500.00	Extreme input accepted
4	Product Price Entry Erroneous Data	£500.[illegible]	[illegible]jects outside of range and displays error message.
5	Product Price Entry Erroneous Data	£3.50	System rejects outside of range and displays error message.
6	Product Price Entry Erroneous Data	"ABS123"	System rejects incorrect format and displays error message.

Acceptance testing

Acceptance testing is performed by the end user on the completed software in its norm testers execute aspects of the planned operation of the system in a realistic and rando dictated by the test plan.

The operations that will be used frequently by the end user are tested more thorough removed prior to the project being signed off.

2.2 – Progress Check

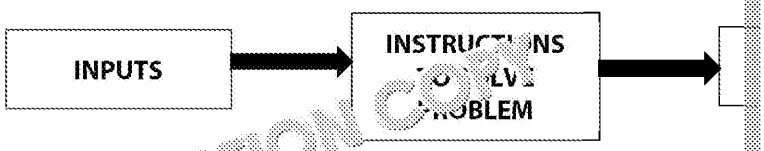
6. (a) List two advantages and two disadvantages of the waterfall method for
   (b) Describe a situation in which the waterfall method is a most appropria
7. (a) Briefly describe the agile methodology (2 marks)
   (b) Describe how pair programming and code reviews are used in extreme
8. (a) Define the term 'prototyping' (1 mark)
   (b) Describe the strengths of using a rapid application development (2 mar
   (c) In which situation is a RAD methodology most suitable? (2 marks)
9. (a) In which situation is a spiral methodology normally used? (2 marks)
   (b) List two advantages and two disadvantage[illegible]e spiral method for so
10. Describe the following:
   (a) White box testing (1 mark)
   (b) Beta testing (2 marks)
   (c) [illegible]ce testing (2 marks)

[illegible]scribe the type of test data that should be used to thoroughly test systen

2.3.1 ALGORITHMS

Analysis and design of algorithms

An **algorithm** is a systematic method based on a series of steps to solve a problem pseudocode to aid understanding rather than in a programming language for exec



Algorithmic develop ed out using the following steps:

- efin problem
- velop a model
- Create an algorithm specification
- Design the algorithm
- Check it operates as expected
- Analyse the performance of the solution
- Implement the algorithm

Many of these points have been covered in the computation thinking part of this publ

Analysis

It is important to analyse the performance of the solution to ensure that it runs efficie

In this section a range of sorting and searching algorithms are implemented and anal a specified task with a given data set.

Implementation

Pseudocode is not program language specific and so cannot be understood by a progr is code written to aid understanding of a problem. Pseudocode needs to be converted it can be executed by a computer.

Well-written pseudocode can be converted into a high-level program language of you

It is suggested that pseudocode should include the following to ensure it is straightfo programming languages.

- The use of meaningful variable names
- Naming procedures and functi g derstandable titles
- Structure pseudo ma g use of white space and indentations to aid un

Sort algorithms

Sort algorithms are normally used to organise data in an array or a list into order. insertion sort techniques will be described, implemented and their suitability for

Bubble sort

In the following algorithm an array 'S' of 'N' elements will be sorted using the bubble

```
BubbleSort(int S[ ], int N)
Swapped ← True
j ← 0
WHILE (Swapped ==True)
 Swapped ← False
       j ← j + 1
 FOR (i=1 TO N-j)
     IF (S[i-1] >  ) THEN
          Temp ← S[i-1]
          S[i-1] ← S[ i ]
          S[ i ] ← Temp
          Swapped ← True
     END IF
 END FOR
END WHILE
```

Refer to the bubb

The algorithm op

- Initially the that when j bubbled) to
- The inside (in the array
- If first elem (S[i]) their
- Once the FO incremente element is

Note that for an a will repeat N-1 ti

Worked example: Sort Array S with six elements (8,9,3,6,2,7)

Bubble sort hand-traced to help explain the process

Note *that for array of six elements outside loop j will repeat five times.*

j	Swapped	8	9	3	6	2	7
1	False	8	9	3	6	2	7
	True	8	3	9	6	2	7
	True	8	3	6	9	2	7
	True	8	3	6	2	9	7
	True	8	3	6	2	7	9
2	True	3	8	6	2	7	
	True	3	6	8	2	7	
	True	3	6	2	8	7	
	True	3	6	2	7	8	
3	False	3	6	2	7		
	True	3	2	6	7		
	False	3	2	6	7		
4	True	2	3	6			
	False	2	3	6			
5	False	2	3				
		2	3	6	7	8	9

Array sorted using

Advantages	D
The bubble sort is a useful tool where there is very little memory required as it only uses the memory which the array or list occupies. It requires a small amount of code and so is simple to understand and implement.	The bubble sort takes a n where N is the size of the because an out-of-positi one position per scan. Co for large data sets.

Insertion sort

In the following algorithm an array 'S' of 'N' elements will be sorted using the insertio

```
InsertionSort (int S[ ], int n)
Integer i, j

FOR (j=2 TO N)
    Key ← S[j]
       i ← j -1

       WHILE (i > 0 AND S[i] > Key)
            S[i+1] ← S[i]   // Swap array
            elements
            i ← i - 1
    END WHILE
       S[i+1] ← Key
END FOR
```

Refer to the insertion so
An insertion sort is an it
follows:

1. Start by taking
   then compare
   number and in
2. Repeat the cor
   fourth number

To summarise, smaller e
the left until they are la
which point they are ins
no elements of input da

Worked example: Trace the stages in sorting the list using an insertion sort with six e

Insertion sort hand-traced to help explain the process
Note that for array of six elements outside loop j will repeat five times.

8	9	3	6	2	7	Original list unsorted	St
8	9	3	6	2	7	9 compared with 8	No
3	8	9	6	2	7	3 compared with 9, 8	3 i
3	6	8	9	2	7	6 compared with,3,8,9	6 i
2	3	6	8	9	7	2 compared with 3,6,8,9	
2	3	6	7	8	9	7 compared with 2,3,6,8,9	

2	3	6	7	8	9	Array sorted usin

Advantages	
The insertion sort is the simplest sort algorithm and is very efficient for small data sets. It is more efficient than the bubble sort as less scans are involved and it is extremely efficient where the data in a list is largely sorted.	The insertion sort nee which is inefficient fo elements is increased decrease.

Search algorithms

A search algorithm is used to find an item with specific properties among a group of it
and linear search techniques will be described, implemented and their suitability for c

Linear search

In this method it is not necessary for the search data to be sorted.

```
Linear Search (Int A[ ], int Target)

Found ← False
FOR (i=1 TO N)
  IF (Target == A[i]) THEN
       OUTPUT "Item Found" Item "A location" i
       Found ← True
       Break    /* Item Found so break from loop
  END IF
FINDER
  IF (Found == False) THEN
  OUTPUT "Item does not exist in array"
END IF
```

Refer to
solution

A linear s
techniqu
search cr
the targe
it has be

If the tar
element
message

Worked example: Use linear search to find the following numbers from the unordered needed to find each number:

(a) 902

(b) 370

(c) 41

1	2	3	4	5	6	7	8	9	10	11	
											Comment
12	14	6	89	41	56	378	370	657	905	902	Data set r
12	14	6	89	41	56	378	370	657	905	902	(a) 902 =
12	14	6	89	41	56	378	370	657	905	902	(b) 370 =
12	14	6	89	41	56	378	370	657	905	902	(a) 41 = 5

Note *that the number of comparisons needed to find the target is directly re*

Binary search

In this method the search data needs to be sorted.

```
Binary Search (Int A[ ], int Target)
Left ← 1
Right ← Size[A]
WHILE (Left <= Right)
    Middle ← (Left + Right) / 2
    IF (A[Middle] = Target) THEN
        Output "Target found at " A[Middle]
    ELSE IF (A[Middle] < Target) THEN
        Left ← Middle + 1
    ELSE
        Right ← Middle – 1
    END IF
END WHILE
```

Refer to the binary sea
A binary search operat
search criteria:

1. First compar
the list
2. Stop if the ta
list
3. If the target
hand list
4. If the target
right-hand li
5. Repeat the p
until the tar

Worked example: Trace the stages of a binary search to find the number 4 from t

1	2	3	4	5	6	7	8	9	10	11	
											Comm
3	4	12	18	21	36	37	49	61	200	702	Data
3	4	12	18	21	36	37	49	61	200	702	INT(1 36 >
3	4	12	18	21	36	37	49	61	200	702	INT(1 12 >
3	4	12	18	21	36	37	49	61	200	702	INT(1 Numb

Note *that the search was successful with three co*

Binary search characteristics

Advantage – the binary search is more efficient than the linear search, as elements can be found with few comparisons, see table on right.

Disadvantage – the data that has to be searched needs to be in order for the binary search.

Total

Stacks

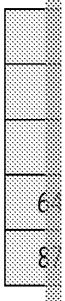
A **stack** is used to as a temporary storage space for data. It is defined as a linear d
in, first out (LIFO) basis.

The stack uses a single pointer (index) to keep track of the data in the stack, where:

- data is inserted (or pushed) onto the stack and
- data is removed (or popped) from the stack.

```
Initially
   Stack-pointer is set to 1 and
   Stack has 'Max' elements
```

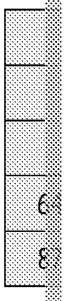
Push Exampl




```
Stack Push Algorithm
     if stack_pointer >= MA
        return sta  ru  l message
     else b
        stack[stack_pointer] = data;
        stack_pointer = stack_pointer + 1;
     end
```

```
Stack Pop Algorithm
     if stack_pointer <= 1 then
        return stack empty message
     else begin
        stack_pointer = stack_pointer - 1;
        data = stack[stack_pointer];
     end
```

Pop Example



Queues

A queue is used as a temporary storage space for data. It is defined as an abstract **in, first out (FIFO)** basis.

The queue uses a two pointer (index) to keep track of the data in the queue, wher

- the start pointer indicates the data item that was first entered into the q
- the end pointer indicates the data item that was last entered into the qu

Assume initially Queue is empty so

Start pointer is set to 0 and

Queue has 'Max' elements

Push Example –

	Start Pointer	
0	1	2
	Ann	Jack

Queue Push Algorithm

```
if start_pointer = 1 and end_pointer =
MAX then
        return queue full message
else if start_pointer = end_pointer + 1
        return queue full message
else begin
        queue [end_pointer] = data;
        end_pointer = end_pointer + 1;
end
```

	Start Pointer	
0	1	2
	Ann	Jack

Queue Pop Algorithm

```
if start_pointer1 then
        return queue empty message
else begin
        data = queue[start_pointer];
        start_pointer = start_pointer + 1
end
```

Pop Example – 'A

→ Start Pointe

0	1	2
		Jack

Suitability of different algorithms for a given task and data set

Sort algorithms

Bubble sort suitability

This method requires very little memory so is useful in embedded systems; however, it is only suitable for small data sets, as it is time-consuming on larger data sets.

This method is ve efficient than the consuming for lar

Search algorithms

Linear search suitability

This method is not very effici th search length can be high; for example if the first item searched for is at the end of the list, all elements of the list have to be searched to find it.

However, the linear search can handle data that is not ordered, and is suitable for small data sets that are unordered.

Binary searching found in relatively

However, the data for applications w sorting would be

2.3 – Progress Check

1. Explain the difference between searching and sorting in computing (2 ma
2. (a) Trace the stages in sorting the list using an insertion sort with seven
   (4 marks)
   (b) Explain why the insertion sort is unsuitable for large data sets (2 ma
3. (a) Briefly describe in words the bubble sort process (2 marks)
   (b) Explain a disadvantage of using the bubble sort approach (2 marks)
4. Describe the method used by the linear search (2 marks)
5. Explain why a binary search method is more efficient than a linear search
6. Trace the stages of a binary search for the number 18 from the list {1,4,6
7. Explain one advantage and one disadvantage of using a binary search te
8. A stack contains the values 7, 6, 9 where 7 is the first value stored and 9
   List the final values in the stack after the following operations are made
   1. Pop
   2. Pop
   3. Push 12
   4. Push 16
   5. Pop
   6. Push 11 (1 mark)
9. A queue contains the values 7, 6, 9 where 7 is the first value stored and 9
   List the final values in the queue after the following operations are made
   1. Pop
   2. Pop
   3. Push 12
   4. Push 16
   5. Pop
   6. Push 11 (1 mark)

ANSWERS

2.1 Elements of computational thinking

2.1.1 (a) Computational thinking (CT) is a problem-solving process that creates a gene
such as problem decomposition, abstraction, modelling and the use of algor
(b) Decomposition or stepwise refinement is an approach that can be used to so
it down into a series of small steps, which can in turn be broken down furthe
(c) Algorithm is a step-by-step approach to solving a problem. It can be written
of any particular programming language (1).
(d) Abstraction is the process of including only the important features when solv
complexity of the system by removing unnecessary details (1).

2.1.2 List the advantages of decomposition
- Breaking the problem into pr[illegible]s to clarify exactly what needs to be
- Each stage of the refinement process creates smaller sub-problems that
- Some functions parts of the solution might be reusable (1)
- [illegible]ki[illegible]n into parts allows more than one person to work on the s

2.1.3 Two b[illegible]s are cache is faster than main memory (1) and it stores regularly use
be processed faster (1). Two drawbacks are cache memory is small compared to R
expensive (1).

2.1.4 A program library is a series of predefined and compiled routines that are availab
a common method for reusing fully working and tested code (1). The use of librar
make the finished program easier to read and understand (1).

2.1.5 Advantages of using sub-procedures are they aid readability making main procedu
possible that sub-procedures can be reused to solve other problems (1). A sub-pr
that has its own task (1) and the development of a solution can be improved as th
developed as a stand-alone task (1).

2.2 Problem-solving and programming

2.2.1 Where local variables are declared and used in a subroutine they are only in existe
executed and are only accessible or in scope within that subroutine (1). It is good p
than global variables in subroutines and functions as it easier to trace the content
and the subroutine retains modularity where only parameters and not global varia

2.2.2 One of the main advantages of using procedures and functions is that less code is
development time will be reduced (1). Also, having less code makes the solution e
maintaining or fixing errors in the program (1).

2.2.3 A parameter is a variable that is used as a data input[illegible]a[illegible]ument that can be u
can be passed by value or by reference. Para[illegible]er [illegible]ed by reference: in this ca
function is a memory reference (1) If an[illegible]e of the variable is changed in the fu
reference will also change (1) wh[illegible]parameter is passed by value, the paramet
the original data (1) [illegible]e of the variable is changed in the function then th
the m[illegible]gr[illegible]

2.2.4 Integra[illegible]development environments (IDE) are widely used to develop software
such as: source code editor (1), debug and error diagnostics (1), run-time environ
automatic documentation facilities (1).

2.2.5 Breakpoints can be added to programs by the debugger; they simply stop the pro
allows the developer to gain knowledge of how the program is operating (1) and t
to locate errors (1).

2.2.6 Waterfall method

(a) *Accept alternatives*

Advantages – ideal for supporting inexperienced project teams and where re
Orderly sequence of development ensures quality documentation with relia
developed software (1).

Disadvantages – dependent upon clear definition of requirements as there i
Produces excessive documentation and keeping it updated as the project pr

(b) This approach is most suitable for the development of mainframe-based or t
(1), where there are clear objectives and no pressure for an immediate imple

2.2.7 Agile methodology

(a) Agile methodologies are a range of alternative methods to traditional softwa
adopts an iterative and incremental method to [illegible]oftware developers res
The agile process is to break down the [illegible]nctional elements and deli
the project duration (1).

(b) Extreme programmi[illegible]

Pair progr[illegible] with this technique two programmers work together at
[illegible]n [illegible] writes the code while the other programmer offers advice on h
[illegible]ing functionality (1).

Code reviews – carried out on a regular or almost continuous basis (1), in sor
pair of programmers on a line-by-line basis to ensure high-quality coding (1)

2.2.8 RAD

(a) Prototyping is an iterative approach based on creating a demonstrable resul
that result. The refinement is based on feedback from the business and the
(b) The production version of an application is available earlier than using other
approach and, therefore, tends to cost less to produce (1). Concentration on
the user with an emphasis on fast completion (1).
(c) RAD methodology is an appropriate method where the requirements of the
the development team is authorised to make design decisions without the n
senior managers (1).

2.2.9 Spiral method

(a) The spiral method is normally used for mission critical projects (1), as well as
(b) Advantages are large amount of risk analysis ensures that issues are address
A software prototype is created early in the life cycle and refined with each s
Disadvantages are a highly skilled development team needed to perform risk
can be high due to the number of prototypes being created and increased cu

2.2.10 Describe the following

(a) This method is based on an internal knowledge of the system to design test
requires programming skills to identify all paths through the software (1).

Exhaustive testing of the system is attempted by: exec[illegible]ing each line of code
statements (1), passing through every branch [illegible]er [illegible]or branch coverage
combinations of true and false conditi[illegible]ns [illegible]e the operation of conditi
(b) User or beta testing involves [illegible]as[illegible]e fully tested software to a limited a
programming team [illegible]at t[illegible]e are few faults or bugs in the system prior t
approach giv[illegible]s [illegible]evelopers the opportunity to view their software
[illegible]of [illegible]ns (1), using real live data can throw up unexpected errors
[illegible]design (1) and integrating in other software environments and with
(c) Acceptance testing is performed by the end user on the completed software
The testers execute aspects of the planned operation of the system in a realis
the sequence dictated by the test plan (1). The operations that will be used fr
more thoroughly, so any faults can be detected and removed prior to the pro

2.2.11 Test data

Normal data – chosen data values must be representative of normal inputs (1).
Extreme data / Boundary data – must be acceptable input data that is on the extr
Erroneous data – this is data that is outside of the normal input range or data tha
expected (1).

2.3 Algorithms

2.3.1 A search algorithm is used to find an item with specific properties among a group algorithm is used to put elements from a list into a specific order (1).

2.3.2 (a) Trace the stages in sorting the list using an insertion sort with seven element

1	8	9	2	5	3	7	Original list unsorted
1	8	9	2	5	3	7	1 compared with 8
1	8	9	2	5	3	7	9 compared with 8,1
1	2	8	9	5	3	7	2 compared with 1,8,9
1	2	5	8	9	3	7	5 compared with 1,2,8,9
1	2	3	5	8	9	7	3 compared with 1,2,5,8,9
1	2	3	5	7	8	9	[illegible] red with 1,2,3,5,8,9

(b) The insertion sort needs a large num[illegible] o[illegible]ent shifts which is inefficient elements is increased the p[illegible]na[illegible]e of the program will decrease (1).

2.3.3 Bubble sort

(a) [illegible] a sort where adjacent items in the array or list are scanned r [illegible]y, until one full scan performs no swaps (1).

(b) The main disadvantage of the bubble sort is that it can take a maximum of (n the size of the list that needs to be sorted (1); this is because an out-of-posit one position per scan (1).

2.3.4 A linear search is the simplest searching technique in the code; the 'target' is the s to compare the target with each element of the list until it has been found (1).

2.3.5 In a linear search, each element in the list is examined until the target value is fou for a large array (1). In a binary search the number of elements being examined is program; for example a maximum of only six comparisons are needed to find a ta

2.3.6 Trace the stages of a binary search for the number 18 from the list {1,4,6,7,9,11,1

1	2	3	4	5	6	7	8	9	10	Index
										Comments
1	4	6	7	9	11	16	18	21	67	Data set number
1	4	6	7	9	11	16	18	21	1	INT(1+10)/2 = Index 5 9 < 18 so choose right
1	4	6	7	9	11	16	18	21	1	INT(1+5)/2 = Index 5 = Number 18 found

2.3.7 An advantage of the binary search is that it is more efficient than the linear search, comparisons, and a disadvantage is that the data that has to be searched needs to

2.3.8 Final content of stack 7, 12, 11 (1)

2.3.9 Final content of queue 12, 16, 11 (1)