UNIT 3

MARK SCHEME

Guidance for examiners

Positive marking

It should be remembered that learners are writing under examination conditions and credit should be given for what the learner writes, rather than adopting the approach of penalising him/her for any omissions. It should be possible for a very good response to achieve full marks and a very poor one to achieve zero marks. Marks should not be deducted for a less than perfect answer if it satisfies the criteria of the mark scheme.

For questions that are objective or points-based the mark scheme should be applied precisely. Marks should be awarded as indicated and no further subdivision made.

For band marked questions mark schemes are in two parts.

Part 1 is advice on the indicative content that suggests the range of computer science concepts, theory, issues and arguments which may be included in the learner's answers. These can be used to assess the quality of the learner's response.

Part 2 is an assessment grid advising bands and associated marks that should be given to responses which demonstrate the qualities needed in AO1, AO2 and AO3. Where a response is not credit worthy or not attempted it is indicated on the grid as mark band zero.

© WJEC CBAC Ltd.

Banded mark schemes

Banded mark schemes are divided so that each band has a relevant descriptor. The descriptor for the band provides a description of the performance level for that band. Each band contains marks.

Examiners should first read and annotate a learner's answer to pick out the evidence that is being assessed in that question. Once the annotation is complete, the mark scheme can be applied.

This is done as a two stage process.

Stage 1 – Deciding on the band

When deciding on a band, the answer should be viewed holistically. Beginning at the lowest band, examiners should look at the learner's answer and check whether it matches the descriptor for that band. Examiners should look at the descriptor for that band and see if it matches the qualities shown in the learner's answer. If the descriptor at the lowest band is satisfied, examiners should move up to the next band and repeat this process for each band until the descriptor matches the answer.

If an answer covers different aspects of different bands within the mark scheme, a 'best fit' approach should be adopted to decide on the band and then the learner's response should be used to decide on the mark within the band. For instance if a response is mainly in band 2 but with a limited amount of band 3 content, the answer would be placed in band 2, but the mark awarded would be close to the top of band 2 as a result of the band 3 content. Examiners should not seek to mark candidates down as a result of small omissions in minor areas of an answer.

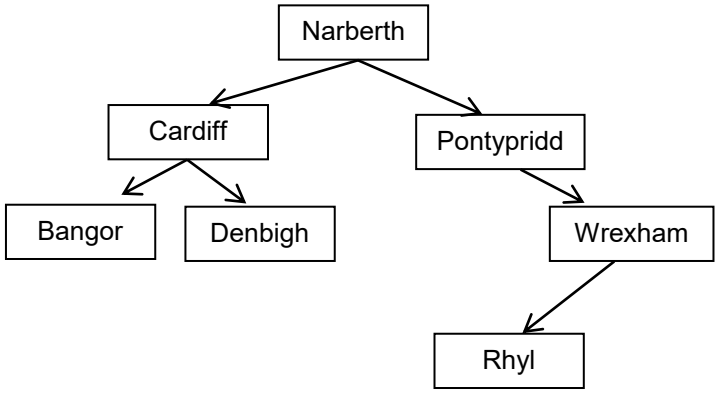
Stage 2 – Deciding on the mark

Once the band has been decided, examiners can then assign a mark. During standardising (marking conference), detailed advice from the Principal Examiner on the qualities of each mark band will be given. Examiners will then receive examples of answers in each mark band that have been awarded a mark by the Principal Examiner. Examiners should mark the examples and compare their marks with those of the Principal Examiner.

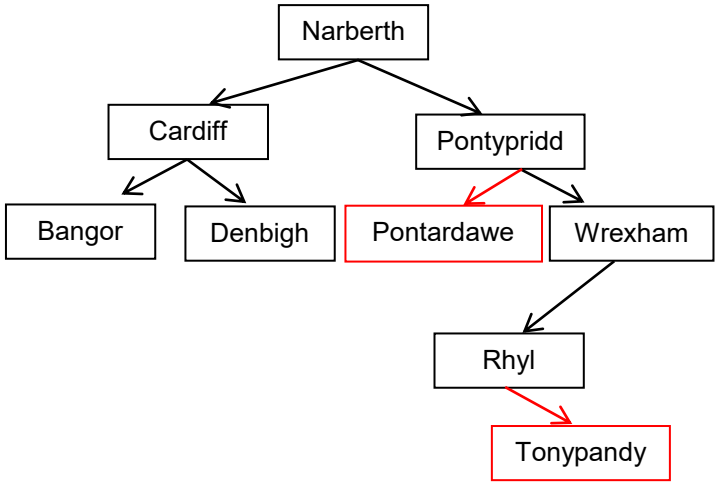
When marking, examiners can use these examples to decide whether a learner's response is of a superior, inferior or comparable standard to the example. Examiners are reminded of the need to revisit the answer as they apply the mark scheme in order to confirm that the band and the mark allocated is appropriate to the response provided.

Indicative content is also provided for banded mark schemes. Indicative content is not exhaustive, and any other valid points must be credited. In order to reach the highest bands of the mark scheme a learner need not cover all of the points mentioned in the indicative content but must meet the requirements of the highest mark band. Where a response is not creditworthy, that is contains nothing of any significance to the mark scheme, or where no response has been provided, no marks should be awarded.

© WJEC CBAC Ltd.

<table>
<tr><th>Q</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>Total</th></tr>
<tr><td>1a</td><td> 

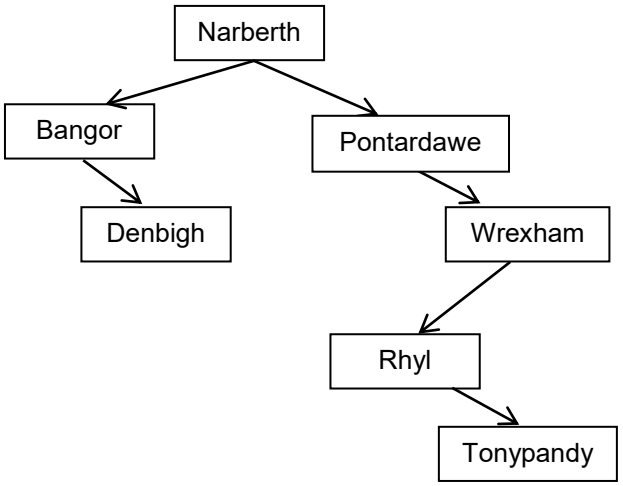
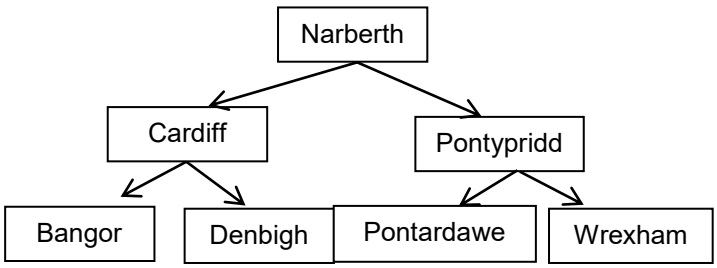
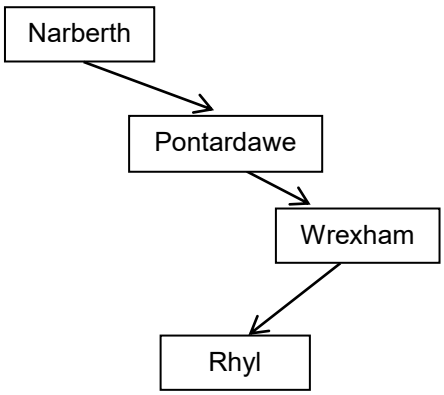
Marking
1 mark for correct root
1 mark for ALL left pointers correct and ALL right pointers correct</td><td>1
1</td><td></td><td>2.1b
2.1b</td><td></td><td>2</td></tr>
<tr><td>1b</td><td>
<table>
<tr><th></th><th>Left Pointer</th><th>Data</th><th>Right Pointer</th></tr>
<tr><td>Start 0</td><td>1</td><td>Narberth</td><td>2</td></tr>
<tr><td>1</td><td>5</td><td>Cardiff</td><td>6</td></tr>
<tr><td>2</td><td>-1</td><td>Pontypridd</td><td>3</td></tr>
<tr><td>3</td><td>4</td><td>Wrexham</td><td>-1</td></tr>
<tr><td>4</td><td>-1</td><td>Rhyl</td><td>-1</td></tr>
<tr><td>5</td><td>-1</td><td>Bangor</td><td>-1</td></tr>
<tr><td>6</td><td>-1</td><td>Denbigh</td><td>-1</td></tr>
<tr><td>7</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td></tr>
</table>

Marking
1 mark for correct root with start pointer
1 mark for ALL left pointers correct
1 mark for ALL right pointers correct</td><td>3</td><td></td><td>2.1a</td><td></td><td>3</td></tr>
<tr><td>1ci</td><td> 
</td><td>1

1</td><td></td><td>2.1b

2.1b</td><td></td><td>2</td></tr>
</table>

© WJEC CBAC Ltd.

1cii	 **Marking** 1 mark for replacing Cardiff with Bangor 1 mark for replacing Pontypridd with Pontardawe 1 mark for removing original Bangor and Pontardawe nodes	1 1 1		2.1b 2.1b 2.1b		3
1d	 The above tree is balanced and the maximum number of comparisons to locate an item would be the same as the number of levels, 3 in this case  The above tree is unbalanced and the maximum number of comparisons to locate an item would be the same as the number of items, 4 in this case **Marking** 1 mark for example of each tree – balanced and unbalanced 1 mark for each description of number of comparisons	2 2		2.1a 2.1b		4
2a	A linked list is a set (accept list) of data elements, where					4

© WJEC CBAC Ltd.

	each element contains: • the data itself • a pointer to the next element Benefit (any one of): • New items can be inserted into a linked list without rearranging all the other elements • If programmed dynamically uses memory more efficiently Drawback (any one of): • A linked list is more complex to program / manipulate than an array • Extra programming is required to access the data in the opposite direction (or the list needs to be doubly linked) • Can only be accessed in a linear manner.	1 1 1 1	1.1b 1.1b 1.1b 1.1b			
2b	<table><tr><th>Address</th><th>Integer</th><th>Pointer</th></tr><tr><td>751</td><td>4811</td><td>756</td></tr><tr><td>752</td><td>2312</td><td>755</td></tr><tr><td>753</td><td>3599</td><td>754</td></tr><tr><td>754</td><td>4166</td><td>751</td></tr><tr><td>755</td><td>2567</td><td>757</td></tr><tr><td>756</td><td>5218</td><td>Null / -1 / 0 / End</td></tr><tr><td>757</td><td>3100</td><td>753</td></tr><tr><td>...</td><td>...</td><td>…</td></tr></table> Marking: 7 correct - 3 marks 5 or 6 correct - 2 marks 3 or 4 correct - 1 mark	3		2.1b		3
3	One mark for each valid reason for following a specific rule, up to a maximum of six marks. No marks for simply stating rules, as question requires reasons for following rules. If rule is implicit within reasoning, award mark for response (e.g. ‘data must be kept safe **to prevent data from being stolen that could cause an individual embarrassment’**) – Security rule implied. Indicative content (bold indicates example valid reasoning within a summary, other valid reasoning accepted, if justified) • Programmers should have due regard for public health, privacy, security and wellbeing of others and the environment. **This will ensure that no harm physical, emotional or financial comes to an individual from not taking these factors into account (could also be reasoned with the use of an appropriate example; for instance in the case of privacy, data could be stolen that could cause an individual embarrassment)** • Programmers should have due regard for the legitimate rights of any person or organisation that might be affected by their activities. **This will ensure that the rights of others are respected and no harm comes to the public (could also be reasoned with the use of an**	6	1.1b			6

© WJEC CBAC Ltd.

	appropriate example; for instance the right of an individual that data held is only used for an agreed purpose and not abused) • Programmers should conduct their professional activities without discrimination on the any grounds. **This will ensure that no individual is denied their rights (could also be reasoned with the use of an appropriate example; for instance that software is carefully designed to consider other groups' needs, for example accessibility for disabled people)** • Programmers should promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise. **This will ensure that there is no technology gap between sectors in society (could also be reasoned with the use of an appropriate example; for instance that a deprived community has opportunity access to the same website as an affluent group)** • Programmers should not claim any level of competence that they do not possess. **This safeguards an employer placing a programmer on a task that could not be completed or would be completed with significant errors which would waste time or money. (could also be reasoned with the use of an appropriate example; for instance a programmer claiming that they could use a given language but could not and then could not write the necessary program)** • Programmers should develop their professional knowledge, skills and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to their field. **This ensures that the product produced by a programmer is up-to-date and will function in contemporary systems (could also be reasoned with the use of an appropriate example; for instance ensuring that a programmer writes software that will function on a new operating system)** • Programmers should ensure that they have the knowledge and understanding of legislation and that they comply with such legislation, in carrying out their professional responsibilities. **This ensures that the programmer does not unwittingly break the law when undertaking their day to day job which could cause embarrassment or losses (could also be reasoned with the use of an appropriate example; for instance developing insecure software that breaches data protection laws)** • Programmers should respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work. **This ensures that all relevant approached and options are considered, and the best one chosen (could also be reasoned with the use of an appropriate example; for instance when developing a user interface all opinions should be considered and the best design used)** • Programmers should avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction. **This ensures that staff are aware that they should consider others before taking action and do not take risks that could injure others (could also be reasoned with the use of an**					

© WJEC CBAC Ltd.

	appropriate example; for instance programmers should avoid altering a program that may lose work for others) • Programmers should reject and not make any offer of bribery or unethical inducement. **This ensures that staff are not open to corruption from others and take actions that could harm a company or client (could also be reasoned with the use of an appropriate example; for instance programmers should not disclose sensitive data if offered an incentive to do so)** • Programmers should carry out their professional responsibilities with due care and diligence in accordance with the employer or client's requirements whilst exercising professional judgement at all times. **This would ensure that programs are developed in line with a client's requirements and that time/money is not wasted in developing other, unrequired areas (could also be reasoned with the use of an appropriate example for instance a programmer should let an employer know if a certain methodology is not working and advise on methods of changing methodology)** • Programmers should seek to avoid any situation that may give rise to a conflict of interest between them and their employer or client. **This would ensure that a programmer does not have conflicting tasks that may result in one not being completed properly (could also be reasoned with the use of an appropriate example; for instance that a programmer should not embark on a personal programming project that competes with that commissioned by a client.)** • Programmers should accept professional responsibility for their work and for the work of colleagues who are defined in a given context as working under their supervision. **This gives ownership of work, and with this, less chance of neglecting the work as the programmer is directly responsible. (could also be reasoned with the use of an appropriate example; for instance if a programmer has a set role in a task, they are likely to feel that they own that task and are more likely to do that task to the best of their ability)** • Programmers should not disclose or authorise to be disclosed, or use for personal gain or to benefit a third party, confidential information except with the permission of their employer or client, or as required by legislation. **This would undermine a client, and possibly result in loss if a competitor were to develop a product based on information disclosed. (could also be reasoned with the use of an appropriate example; for instance this prevents a programmer from selling information on a product to a company developing a similar product)** • **Programmers should not** misrepresent or withhold information on the performance of products, systems or services (unless lawfully permitted to do so by a duty of confidentiality) or take advantage of the lack of relevant knowledge or inexperience of others. **This prevents making financial or other gain from overstating the work required for a given task. (could also be**					

© WJEC CBAC Ltd.

	reasoned with the use of an appropriate example; for instance could mean that a programmer could charge more money by stating that a simple task took longer to complete than it actually did) • Programmers should accept their personal duty to uphold the reputation of the profession and not take any action which could bring the profession into disrepute. **This ensures that the profession is not seen negatively by the wider public and not undermined by a lack of trust. (could also be reasoned with the use of an appropriate example – many potential examples)** • Programmers should encourage and support fellow members in their professional development. **This ensures that fellow members are able to support their team in development and that individuals are not undermined or lose out as a result of a lack of knowledge. (could also be reasoned with the use of an appropriate example – many potential examples)** • Programmers seek to improve professional standards through participation in their development, use and enforcement. **This ensures that programmers have ownership of the standards and these standards are more likely to be relevant to programmers as a result. (could also be reasoned with the use of an appropriate example, for instance if there were a new programming certification, the fact that programmers were part of its development would give the certification more status)** • Programmers notify the employer if convicted of a criminal offence. **This ensures that an employer can judge if a programmer can continue in their role as there may be risks if the crime is relevant to their work. (could also be reasoned with the use of an appropriate example – for instance if convicted of fraud, a programmer would not be permitted to program financial systems)**					
4a	$(A + B).(A + \overline{B})$ $A.A + A.\overline{B} + B.A + B.\overline{B}$ $A + A.\overline{B} + B.A$ A	 1 1 1		 2.1a 2.1a 2.1a		3
4b	$C + \overline{B}\overline{C}$ $C + (\bar{B} + \bar{C})$ $(C + \bar{C}) + \overline{B}$ $1 + \overline{B}$ 1	 1 1 1		 2.1a 2.1a 2.1a		3
4c	$A + (A + \overline{B.C}) + \overline{C}$ $A + (A + \overline{B} + \overline{C}) + \overline{C}$ $A + A + \overline{B} + \overline{C} + \overline{C}$ $A + \overline{B} + \overline{C}$	 1 1 1		 2.1a 2.1a 2.1a		3
5	X 10011011_2 Y 11010111_2 **XOR 01001100_2** **Retrieving the original** 01001100_2 Y 11010111_2 **XOR 10011011_2**	1 1		2.1a 2.1a		3

© WJEC CBAC Ltd.

	The encrypted data is produced by “XOR-ing” the actual data (X) with the key (Y). The resulting encrypted data can only be read by someone who knows the key. This decryption is achieved by XOR-ing the encrypted data again using the same key (Y)to obtain the original data Accept reversed solution (XOR of Y with X).	1		1.1b		

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
6	1 set SearchArray(0 to n-1) 2 set Start = 0 3 set End = n-1 4 set Found = False 5 6 input SearchValue 7 8 repeat 9 set Mid = (Start + End) DIV 2 10 if SearchValue = SearchArray(Mid) then 11 set Found = True 12 Output "SearchValue found at position", Mid 13 endif 14 15 if SearchValue > SearchArray(Mid) then 16 set Start = Mid + 1 17 endif 18 19 if SearchValue < SearchArray(Mid) then 20 set End = Mid – 1 21 endif 22 until (Found = True) OR (End < Start) 23 24 if Found = False 25 Output "SearchValue not found" 26 endif **Marking** • Declare array and initialise variables • Loop structure + increment • Calculate + output position if found • Correct terminating condition for loop • Correctly discard half of array if SearchArray(Mid) > SearchValue • Correctly discard half of array if SearchArray(Mid) < SearchValue • Output message if not found	1 1 1 1 1 1 1			3.1b 3.1b 3.1b 3.1b 3.1b 3.1b 3.1b	7
7	Name: Recursive algorithm Features: • Must also have a terminating condition (base case / stopping condition) • A recursive algorithm is one which calls itself Example sort: Quicksort	1 1 1 1	 1.1b 1.1b 1.1a	2.1a		4

© WJEC CBAC Ltd.

Q	Answers	Marks	AO1	AO2	AO3	Total
8a	BNF is used to describe (unambiguously) the syntax / grammar / rules of a programming / computer language	1	1.1b			1
8bi	<digit> ::= 0|1|2| ... |9 <letter> ::= A|B|C| ... |Z <numeric> ::= <digit><digit><digit><digit><digit><digit> <description> ::= <letter>|<letter><description> <component code> ::=<numeric><description> **Marking:** 1 mark for recursion Same item Left and Right are needed Cannot gain full marks unless completely correct Incorrect notation – deduct 1 mark	1 1 1 1		2.1b 2.1b 2.1b 2.1b		4
8bii	digit → digit → digit → digit → digit → digit → letter → **Marking:** 1 mark for digit x 6 (lose if "sixdigits" single box) 1 mark for letter 1 mark for recursion	 1 1 1		 2.1b 2.1b 2.1b		3
9a	All three must be correct $10^2 = 100$ $100^2 = 10,000$ $1000^2 = 1,000,000$	 1		 2.1a		1
9b	Evaluation of algorithm Comparison The only comparison appears in the j loop. Since this loop will iterate a total of n^2 times, it will execute exactly n^3 comparisons Data swap There may be a swap operation carried out in the j loop. Swap(A[i-1], A[i]) Each of these will happen n^2 times. Therefore there are $2n^2$ operation carried out within the j loop The i loop has one addition operation incrementing i which happens n times Adding these up we the number of addition operations which is $2n^2 + n$ As n gets very big then n^2 will dominate therefore it is $O(n^2)$ NOTE: Calculations might include assignment operations but these will not affect overall time so ignore **Marking:** 1 mark for identifying i loop will execute n times. 1 mark for identifying j loop will execute $2n^2$ times. 1 mark for correct number of calculations $2n^2 + n$ 1 mark for determining that the order will be dominated by n^2 as n gets very big giving $O(n^2)$ for the algorithm	1 1 1 1			3.1c 3.1c 3.1c 3.1c	4

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
9c	Algorithm will need to store 1 array that will require n elements. The total storage will therefore be 1 x n As n increases the storage requirements will increase by n as constant (1) will be insignificant so storage requirements will be Order (n)	1 1			3.1c	2
10	**The Waterfall approach** • Sequential design process, in which various developers draft up all of the requirements for a system up front **Advantages (any one of)** • By having all the requirements beforehand, everyone knows exactly what's needed • Client knows what to expect, including time frame, size, and cost of the project, and they know exactly what their product will do • If employees leave or join the development team, the strong documentation allows bringing new people up to speed quickly **Disadvantages (any one of)** • Because the process is sequential, once a stage of development has been completed, you can't go back to a previous stage to make changes • If the initial requirements of the project are faulty in any way, the project is almost guaranteed to fail • The product is only tested once it is completed and if bugs were made early on, a large amount of code will be affected • If the client's needs change as the project goes on, the project will take longer than predicted **The Agile approach** • Incremental approach to development, in which developers start off with a simple project design instead of a huge document, and work on small modules at a time **Advantages (any one of)** • Changes can be made after the initial planning phase, and as the client makes changes the program can be re-written • Testing is done as the product is developed, ensuring that bugs are found earlier in the process • A smaller team can work on the product because you are removing the upper layers of project managers • There can be a closer relationship between the customer and the developer • When the end goal of the product is not clearly defined, Agile development is the most suitable approach • Sprints of work on the project are done and priorities of the project are discussed, evaluated, and tested Then, a simple product is released to the consumer and they are now able to use it and provide feedback **Disadvantages (any one of)** • It can be hard to employ new people into a team when you have less of a clearly defined structural process • It can be difficult to predict when the project will be completed, or how much it will ultimately cost.	1 1 1 1 1 1	1.1b 1.1b 1.1b 1.1b 1.1b 1.1b			6

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
11a	• Check the correspondence between the actual design and its specification / user requirements / objectives / safety issues • Confirm that the most appropriate techniques have been used • Confirm the HCI is appropriate	1 1 1	1.1b 1.1b 1.1b			3
11b	If programmer A modifies current version, and programmer B modifies an earlier version, neither new version will contain both modifications	1	1.1b			1
12a	**Assemblers** • An assembler converts the low level assembly programming language into machine code. **Interpreters** • An interpreter converts high level language code one line at a time, into machine code, which is then executed by the CPU. **Compilers** • A compiler translates the entire high level programming language (source code) into machine code programs prior to execution.	1 1 1	1.1b 1.1b 1.1b			3

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
12b	1 mark for naming an error x 3 1 mark for describing that error x 3 1 mark for each example x 3 **Syntax** A syntax error occurs when a command does not follow the expected syntax of the language. For instance, when a keyword is incorrectly spelt. Error • Incorrect: IF A ADN B Then • Correct: IF A AND B Then **Runtime / Execution** A runtime error is an error that only occurs when the program is running and is difficult to foresee before a program is compiled and run. Error: • Program requests more memory when none is available so the program crashes. **Logical** A logical error is an error that causes a program to output an incorrect answer (that does not necessarily crash the program). Error: • An algorithm that calculates a person's age from their date of birth but ends up giving negative numbers. **Linking** A linking error occurs when a programmer calls a function within a program and the correct library has not been linked to that program. Error: • When the Square-Root function is used and the library that calculates the Square-Root has not been linked to the program. **Rounding** Rounding is when a number is approximated to nearest whole number / tenth / hundredth, etc. Error: • 34.5 rounded to nearest whole number is 35, an error of +0.5. **Truncation** Truncating is when a number is approximated to a whole number / tenth / hundredth, etc. nearer zero. Error: • 34.9 truncated to whole number is 34, an error of-0.9.	3 3 3	1.1b 1.1b 1.1b			9

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
13	**Indicative content** **<u>Procedural languages</u>** • Procedural languages are used in traditional programming based on algorithms or a logical step-by-step process for solving a problem • They obey (ordered) instructions • They carry out actions / calculations etc. • A procedural programming language provides the programmer a way to define precisely each step when performing a task • Allows tight control over the underlying operation of the hardware • Used in (large complicated) programs where similar operations may be carried out at varying stages of the program execution **<u>Scripting Language</u>** • Set of commands understood by the applications software • Usually embedded in another language and is used to control aspects of the software • Usually a High-level programming language • Can be interpreted not compiled • Scripting languages provides the programmer a way to define precisely each step when performing a task • Allows tight integration with existing programs or data • Script embedded in (the HTML in) a web site to control graphics, etc. • Script embedded in a web site to load / execute a file when clicked, etc. **<u>Non-Procedural languages</u>** • Non-procedural programming languages allow programmers to specify the results they want without specifying how to solve the problem • Non-procedural languages are to do with rules / making queries • Used in database interrogation where retrieving answers are more important than the exact steps required to calculate the result • Artificial intelligence and modelling applications are often written in a non-procedural language **<u>Object Orientated Language</u>** • Uses objects and classes - include both data and associated processing • Applies the principles of encapsulation, inheritance and polymorphism to aid programming • Enables production of buttons / icons etc. - useful in a visual environment • A class defines the methods and properties (data) for a group of similar objects	13	1.1b			13

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
	• Once an object is created, knowledge of its implementation is not necessary for its use. • Objects control how other objects interacts with themselves, preventing other kinds of errors, e.g. a programmer cannot set the width of a window to -500 • In Visual Basic, the programmer places objects on forms. It is an event-driven language • An event, e.g. click a command button, initiates a sequence of code to be executed • Objects created using object oriented languages can easily be reused in other programs **Special Purpose Language** • Languages that were designed with a specific purpose in mind as opposed to a more general use language • Might have essential / helpful features relevant to the application • Are available for simulation, control etc. • Very specialised with built in functions/abilities that lend themselves directly to solving the problem that the language was design to work on • Used in: o Computer aided design o Artificial intelligence o Expert systems o Scientific applications o Games programming (DirectX etc) **4th Generation Language** • First generation programming languages created construct above the machine-code program • Each subsequent generation represented a further distancing from the binary code that the computer hardware actually reads • Some packages, e.g. Microsoft Access, have in-built programming capabilities. This allows the programmer to customise general purpose packages to exactly meet the needs of the business. • Generally a very high level programming language (English syntax and grammar) • Many features such as query, manipulation of data • May have report generators and possibly application generators • May attempt to produce natural language interface • Requires less programming skill • Would be useful in a database query / manipulation situation • Often used in conjunction with end user applications to customise their operation without requiring highly developed and specialised programming skills					

© WJEC CBAC Ltd.

Q	Answer	Marks	AO1	AO2	AO3	Total
	Natural Languages • The user would not need to structure voice (or typed) input in any way - could communicate with the computer as if with another person • A natural language interface would need very high processing power / very complex software • Natural language used by most people is very ambiguous / imprecise / doesn't tend to conform to set grammar / slang is often used / English language is changing • Symbolic languages are capable of interpreting and processing queries by sentences, e.g. calculating mathematical equations • Allows the user to speak in their normal everyday language in order to interact with the computer • Speak everyday commands, such as “Open the last document I used” **Visual Programming Languages** • High level programming language • Particularly suitable for production of objects / buttons / icons, etc. • Particularly suitable for developing in a GUI / graphics content / event driven environment (e.g. double-click > execute) • May be easier to learn / more intuitive because visual / very good help / tools available **Application packages that have programming capabilities** • Additional functionality can be added without a programmer / buying another package / program • Can customise the package / tailor to specific needs etc • Requires less / no programming skill > more help is available in the package • Is probably cheaper / quicker since most facilities are provided by the package • Can import / export from / to other packages • Is less likely to contain errors “bugs” / package has already been well tested • Users are probably familiar with interface • Programming might be restricted and have certain functionality unavailable in the package					

© WJEC CBAC Ltd.

Band	AO1.1b Max 13 marks
3	**10 - 13 marks** The candidate has: • written an extended response that has a sustained line of reasoning which is coherent, relevant, and logically structured • shown clear understanding of the requirements of the question and a clear knowledge of the indicative content. Clear knowledge is defined as a response that provides ten to thirteen relevant detailed points on the nature of different programming paradigms, which relate to an extensive amount of the indicative content • addressed the question appropriately with minimal repetition and no irrelevant material • has presented a balanced discussion and justified their answer with examples • used appropriate technical terminology referring to the indicative content confidently and accurately.
2	**5 - 9 marks** The candidate has: • written a response that has an adequate line of reasoning with elements of coherence, relevance, and logical structure • shown adequate understanding of the requirements of the question and a satisfactory knowledge of the topic of changeover as specified in the indicative content. Satisfactory knowledge is defined as a response that provides five to nine points on the nature of different programming paradigms as signalled in the indicative content. • has presented a discussion with limited examples • used appropriate technical terminology referring to the indicative content.
1	**1 - 4 marks** The candidate has: • written a response that that lacks sufficient reasoning and structure • produced a discussion which is not well developed • attempted to address the question but has demonstrated superficial knowledge of the topics specified in the indicative content. Superficial knowledge is defined as a response that provides one to four points on the nature of different programming paradigms(or a single paradigm) as signalled in the indicative content • used limited technical terminology referring to the indicative content.
0	**0 marks** Response not credit worthy or not attempted.

© WJEC CBAC Ltd.