GCSE

Computer Science

Unit 2 – Computing Fundamentals

Mark scheme

4512/2

June 2016

Version: 1.1 Final

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Assessment Writer.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from aqa.org.uk

<table>
<tr><th>Qu</th><th>Part</th><th>Sub-part</th><th>Marking Guidance</th><th>Marks</th></tr>
<tr><td>1</td><td>a</td><td></td><td>101 0111;

I. Leading zeros</td><td>1</td></tr>
<tr><td>1</td><td>b</td><td></td><td>1100 1110;

If answer given is 11001110 then reward any attempt at working;
If the answer given is not 11001110 then a maximum of 1 mark can be awarded for any of the following working out stages:

• C or E (but not both) are converted to an incorrect binary representation but are then combined with the other correct representation. For example C is converted incorrectly to 1001 but E is converted correctly to 1110 and the answer given is 10011110;
• C is converted to a denary number other than 12 and/or E is converted to a denary number other than 14 but both of the denary numbers are correctly converted to binary.
• The candidate has attempted to multiply 16 by 12 and 1 by 14 but has then incorrectly converted the result into binary (through either an initial multiplication error or binary conversion error but not both).</td><td>2</td></tr>
<tr><td>1</td><td>c</td><td></td><td>1 mark for one correct row;
Both marks for all three correct rows;

<table><tr><th>Number</th><th>Order (1 – 3)</th></tr><tr><td>The denary number 12</td><td>3</td></tr><tr><td>The binary number 1110</td><td>1</td></tr><tr><td>The hexadecimal number D</td><td>2</td></tr></table>
R. if duplicate numbers have been used</td><td>2</td></tr>
<tr><td>1</td><td>d</td><td></td><td>5;</td><td>1</td></tr>
<tr><td>1</td><td>e</td><td></td><td>The day, month and year;
could each be stored as separate values (within a data structure);

A. any suitable data type instead of values; (within a record)</td><td>2</td></tr>
<tr><td>1</td><td>f</td><td>i</td><td>It has a larger storage capacity / it can hold more data;</td><td>1</td></tr>
</table>

<table>
<tr><td>1</td><td>f</td><td>ii</td><td>Any creditworthy point to a maximum of two. Examples of typical advantages of solid state over magnetic storage include:

It has a higher read/write speed;
It is smaller;
It is more robust;
It generates less heat;
It has a lower power consumption;
It is lighter;
It is quieter;</td><td>2</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>a</td><td></td><td>Information is data with context // meaning;

A. Information is data that has been processed;</td><td>1</td></tr>
<tr><td>2</td><td>b</td><td></td><td>1 and 0 could represent true and false // A bit and a Boolean data type both have only two values;

A. other wording that has equivalent meaning.</td><td>1</td></tr>
<tr><td>2</td><td>c</td><td></td><td>8;</td><td>1</td></tr>
<tr><td>2</td><td>d</td><td></td><td>7;

A. 8-bits; (extended ASCII)</td><td>1</td></tr>
<tr><td>2</td><td>e</td><td></td><td>1 mark for each correct label;

The correct table is:

<table>
<tr><th>Value</th><th>Label (A-E)</th></tr>
<tr><td>43.13</td><td>C</td></tr>
<tr><td>“Curry-Howard”</td><td>E</td></tr>
<tr><td>978</td><td>A</td></tr>
</table>

A. Real instead of C, string instead of E and integer/Int instead of A

R. If a mark is duplicated

For example, this answer would score 1 mark:

<table>
<tr><th>Value</th><th>Label (A-E)</th></tr>
<tr><td>43.13</td><td>C</td></tr>
<tr><td>“Curry-Howard”</td><td>A</td></tr>
<tr><td>978</td><td>A</td></tr>
</table>
</td><td>3</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>And this answer would score two marks:

<table><tr><th>Value</th><th>Label (A-E)</th></tr><tr><td>43.13</td><td>C</td></tr><tr><td>“Curry-Howard”</td><td>E</td></tr><tr><td>978</td><td>B</td></tr></table>
And this answer would score zero marks:

<table><tr><th>Value</th><th>Label (A-E)</th></tr><tr><td>43.13</td><td>C</td></tr><tr><td>“Curry-Howard”</td><td>C</td></tr><tr><td>978</td><td>C</td></tr></table></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>a</td><td>i</td><td>Fourth box only;

<table><tr><th>Test data</th><th>Tick one box</th></tr><tr><td><code>-1, 0, 9, 10</code></td><td></td></tr><tr><td><code>0, 1, 10, 11</code></td><td></td></tr><tr><td><code>-1, 0, 10, 11</code></td><td></td></tr><tr><td><code>0, 1, 9, 10</code></td><td>✓</td></tr></table></td><td>1</td></tr>
<tr><td>3</td><td>a</td><td>ii</td><td>They test the boundaries;

A. other wording that has equivalent meaning.</td><td>1</td></tr>
<tr><td>3</td><td>b</td><td></td><td>Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled A – E and shown in the examples where they are awarded:

A. 1 mark for assigning user input to a variable (permit any variable name);
B. 1 mark for assigning the second user input to a distinct variable from that used in A, or the second user input used in such a way that doesn’t require a variable, e.g. USERINPUT = password1;
C. 1 mark for using a condition-controlled loop such as a <code>WHILE</code> loop with a correct Boolean expression to control this loop (this will depend on the type of loop used but will test for equality of the two passwords);
D. 1 mark for the user inputting the two passwords again within the loop and assigning these to their respective variables;
E. 1 mark for outputting “<code>password created</code>” at a point in the code where no further user input will occur
(A. spelling mistakes for “<code>password created</code>”);</td><td>5</td></tr>
</table>

Example 1 (italicised square brackets indicate where marks are awarded):

```
password1 ← USERINPUT [A]
password2 ← USERINPUT [B]
WHILE password1 ≠ password2 [C]
  password1 ← USERINPUT
  password2 ← USERINPUT [D with line above]
ENDWHILE
OUTPUT 'password created' [E]
```

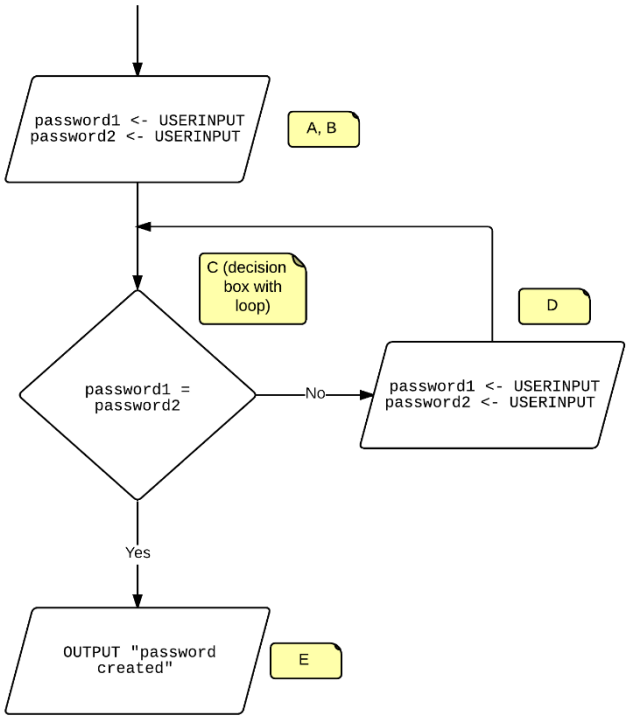
Example 2 (italicised square brackets indicate where marks are awarded):

```
password1 ← USERINPUT [A]
password2 ← USERINPUT [B]
match ← false
IF password1 = password2 THEN
  match ← true
ENDIF
WHILE match = false [C with three lines above and the IF statement within the loop]
  password1 ← USERINPUT
  password2 ← USERINPUT [D with line above]
  IF password1 = password2 THEN
    match ← true
  ENDIF
ENDWHILE
OUTPUT 'password created' [E]
```

Example 3 (italicised square brackets indicate where marks are awarded):

```
match ← false
REPEAT
  password1 ← USERINPUT [A]
  password2 ← USERINPUT [B][D with line above]
  IF password1 = password2 THEN
    match ← true
    OUTPUT 'password created' [E]
  ENDIF
UNTIL match = true [C for condition for DO-WHILE]
```

Example 4 (notes indicate where marks are awarded):

				
3	**c**		Any creditworthy point to a **maximum of two**. Examples include: They do not have a minimum length//they can be blank; They do not have to have a mixture of alphabetic and numeric characters; They may have consecutive numbers within them; They may be easily guessable names;	**2**
4	**a**	**i**	Star; **A.** Switched Ethernet	**1**
4	**a**	**ii**	Any creditworthy point to a **maximum of two**. Examples include: It allows each client to use full transmission over the network // no data collisions; Easy to connect new devices; The failure of one client will not affect the others; Packets are only sent to intended device; Improved performance; (compared to non-switched networks)	**2**

<table>
<tr><td>4</td><td>b</td><td></td><td>1 mark if 1 action correct;
2 marks if 2 actions correct;
3 marks if all 4 actions correct;

The correct table is:

<table><tr><th>Action</th><th>Order (1 – 4)</th></tr><tr><td>The client receives the webpage</td><td>4</td></tr><tr><td>The client requests the webpage</td><td>1</td></tr><tr><td>The server delivers the webpage</td><td>3</td></tr><tr><td>The server connects to a database to complete the webpage</td><td>2</td></tr></table></td><td>3</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>a</td><td></td><td>Any creditworthy points to a maximum of three. Examples include:

The testing was not exhaustive // did not cover all paths through the program;
The action taken after testing was incorrect // contained other errors;
The error only occurs very infrequently;
Software updates may introduce new errors;
Badly designed test strategy;
Code very complex;
Poor understanding of system requirements.</td><td>3</td></tr>
<tr><td>5</td><td>b</td><td></td><td>Any creditworthy point to a maximum of one. Examples include software controlling:

Nuclear reactors;
Transport signalling;
Vehicle braking;
Air traffic control;</td><td>1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>a</td><td></td><td><table><tr><th>Most suitable data type of n</th><th>Tick one box</th></tr><tr><td>String</td><td></td></tr><tr><td>Boolean</td><td></td></tr><tr><td>Integer</td><td>✓</td></tr></table></td><td>1</td></tr>
<tr><td>6</td><td>b</td><td></td><td>A function returns a value (whereas a procedure does not);</td><td>1</td></tr>
<tr><td>6</td><td>c</td><td></td><td>1 mark for each correct row;

<table><tr><th>Statement</th><th>Tick two boxes</th></tr><tr><td><code>Display(2, 6)</code> and <code>Display(2, 3)</code></td><td>✓</td></tr></table></td><td>2</td></tr>
</table>

				will both have the same output.		
				`Display(2, 6)` and `Display(6, 2)` will both have the same output.		
				`Display(2, -1)` will not output anything.	✓	
				`Display(-2, 1)` will output two different values.		

<table>
<tr><td>6</td><td>d</td><td></td><td>1 mark for the first value of 1;
1 mark for incrementing each row by 1;
1 mark for the last value of 4;
1 mark for outputs being twice the value of x if at least two values of x are given (I. if x is incorrect)

The completed correct trace table is:

<table><tr><th>x</th><th>Output</th></tr><tr><td>1</td><td>2</td></tr><tr><td>2</td><td>4</td></tr><tr><td>3</td><td>6</td></tr><tr><td>4</td><td></td></tr></table></td><td>4</td></tr>
<tr><td>6</td><td>e</td><td></td><td>1 mark for starting at 3;
1 mark for outputting 6 and nothing further (except 3);</td><td>2</td></tr>
<tr><td>6</td><td>f</td><td></td><td>Any creditworthy points to a maximum of two. Examples include:

They can be tested in isolation;
They only need to be tested once;
They can be more easily updated;
They make code easier to understand (to a human);
Likely to reduce number of lines of code in a program;</td><td>2</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td><table><tr><td>No creditworthy material</td><td>0</td></tr><tr><td>Lower mark range

Syntax and run-time errors are weakly described (or described by example) with no comparison between the two.
//
A brief attempt at comparison is made with contains at least one correct point but it may contain some errors.
//
Only one of the errors is well described.

Quality of written communication: The candidate has used a form and style of writing which has many deficiencies. Ideas are not often clearly expressed. Sentences and paragraphs are often not well-connected or at times bullet points may have been used. Specialist vocabulary has been used inappropriately or not at all. Much of the text is legible and some of the meaning is clear. There are many errors of spelling, punctuation and grammar but it should still be possible to understand much of the response.</td><td>1-2 marks</td></tr></table></td><td>6</td></tr>
</table>

				Mid mark range Syntax and run-time errors are both correctly described and their differences are stated/there is limited attempt at comparison. Quality of written communication: The candidate has mostly used a form and style of writing appropriate to purpose and has expressed some complex ideas reasonably clearly and fluently. The candidate has usually used well linked sentences and paragraphs. Specialist vocabulary has been used on a number of occasions but not always appropriately. Text is legible and most of the meaning is clear. There are occasional errors of spelling, punctuation and grammar.	3-4 marks		
				High mark range Syntax and run-time errors are correctly described and their differences are explained. Quality of written communication: The candidate has selected and used a form and style of writing appropriate to purpose and has expressed complex ideas clearly and fluently. Sentences and paragraphs follow on from one another clearly and coherently. Specialist vocabulary has been used appropriately throughout. Text is legible and the meaning is clear. There are few if any errors of spelling, punctuation and grammar.	5-6 marks		
				Quality of written communication skills The candidate's quality of written communication skills will be one of the factors influencing the actual mark an examiner will give within a level of response. The quality of written communication skills associated with each level is indicated above.			
				Description of a syntax error • When a program cannot be understood because it does not follow the rules of the language.			

			Description of a run-time error • Errors that occur during the execution of a program as a result of mistakes in the program (or other external factors). **Potential comparisons between syntax and run-time errors:** • Syntax errors are errors in the program code which break the rules of the language whereas runtime errors are written as syntactically correct code. • Syntax errors are normally picked up early in development whereas run-time errors may only appear late (or never) in testing. • IDE tools will help to identify syntax errors but are less useful in identifying run-time errors. Syntax errors are often easier to locate and fix. • Syntax errors will mean that a program cannot run whereas a run-time error will cause a running program to crash. • Run-time errors may not occur every time the program is run whereas syntax errors will prevent a program running until they are fixed.			
8			No creditworthy material	0		**6**
			Lower mark range Vague statements are made about how clock speed and/or one other characteristic can affect CPU performance // Clock speed not mentioned but another CPU characteristic is described Quality of written communication: The candidate has used a form and style of writing which has many deficiencies. Ideas are not often clearly expressed. Sentences and paragraphs are often not well-connected or at times bullet points may have been used. Specialist vocabulary has been used inappropriately or not at all. Much of the text is	1-2 marks		

				legible and some of the meaning is clear. There are many errors of spelling, punctuation and grammar but it should still be possible to understand much of the response.			
				Mid mark range Clear descriptions are made about how clock speed affects performance. One other CPU characteristic is described. Quality of written communication: The candidate has mostly used a form and style of writing appropriate to purpose and has expressed some complex ideas reasonably clearly and fluently. The candidate has usually used well linked sentences and paragraphs. Specialist vocabulary has been used on a number of occasions but not always appropriately. Text is legible and most of the meaning is clear. There are occasional errors of spelling, punctuation and grammar.	3-4 marks		
				High mark range A correct and detailed explanation of how clock speed affects CPU performance is given, along with a correct and detailed description of one other CPU characteristic and its effect on performance. Quality of written communication: The candidate has selected and used a form and style of writing appropriate to purpose and has expressed complex ideas clearly and fluently. Sentences and paragraphs follow on from one another clearly and coherently. Specialist vocabulary has been used appropriately throughout. Text is legible and the meaning is clear. There are few if any errors of spelling, punctuation and grammar.	5-6 marks		
				Quality of written communication skills The candidate's quality of written communication skills will be one of the factors influencing the actual mark an examiner will give within a level of response. The quality of written communication skills associated with each level is indicated above.			

			Explanation of clock speed Instructions are fetched from memory; Decoded//Executed by the processor; The speed at which this cycle happens; Is directly related to the clock speed; So a higher clock speed means more instructions can be executed (per unit time). **Description of other characteristics may include:** **Cache memory** Frequently used instructions/data; Instructions/data which is predicted to be used; Are pre-loaded into cache; Which is faster to access than RAM/main memory; Is located on or close to the processor; Reduces the time to fetch data/instructions; **Number of cores** One processor/CPU has multiple cores; Each core can process instructions independently of the other; Allow more than one instruction/process to be processed in parallel;	
9	**a**	**i**	6;	**1**
9	**a**	**ii**	(Because it is) not unique; **A.** Answer given by correct example	**1**
9	**a**	**iii**	Foreign key // forms a relationship/link between the tables;	**1**

<table>
<tr><td>9</td><td>b</td><td></td><td>Holly, Faluyi
Chloe, Smith
Ella, Williams

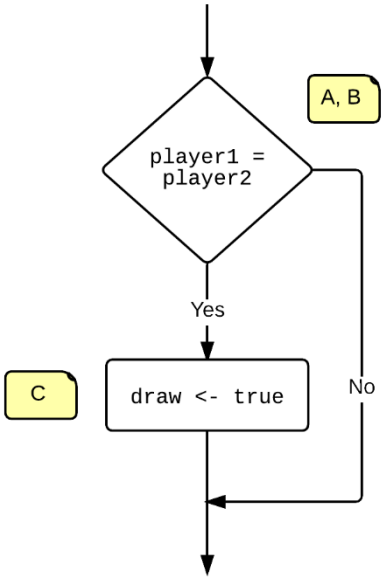
1 mark for only displaying the Firstname field followed by the Lastname field (do not award if less or more fields are given or the order of these two fields is incorrect);
1 mark for at least two of the three correct records (also permit Max Taylor and John Jones in addition);
1 mark for only identifying the three correct records (no more and no less);
1 mark for results displayed in ascending alphabetical order on the last name of the student (permit if other fields incorrect as long as there are at least two records shown);

I. missing commas</td><td>4</td></tr>
<tr><td>9</td><td>c</td><td></td><td>The information does not update (automatically);
//
The information has to be updated regularly;
//
Data can become inaccurate;
//
It is not as accurate as a date of birth;</td><td>1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>a</td><td></td><td>Third box only;

<table><tr><th>Line of code</th><th>Tick one box</th></tr><tr><td>IF player1 = 1 OR player2 = 2 THEN</td><td></td></tr><tr><td>IF player1 ≠ player2 THEN</td><td></td></tr><tr><td>IF player1 = 1 AND player2 = 2 THEN</td><td>✓</td></tr></table></td><td>1</td></tr>
<tr><td>10</td><td>b</td><td></td><td>Array // list;

A. correct programming language-specific data structure
R. String</td><td>1</td></tr>
<tr><td>10</td><td>c</td><td></td><td>Correct row only;

<table><tr><th>Programming technique</th><th>Tick one box</th></tr><tr><td>Iteration</td><td>✓</td></tr><tr><td>Selection</td><td></td></tr><tr><td>Variable assignment</td><td></td></tr></table>
R. more than one row ticked</td><td>1</td></tr>
</table>

10	**d**		Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled **A – C** and shown in the examples where they are awarded: **A.** 1 mark for using selection; **B.** 1 mark for a Boolean condition that tests for equivalency of the variables `player1` and `player2`; **C.** 1 mark for assigning the value `true` to the variable `draw`; **Example 1** (italicised square brackets indicate where marks are awarded): `IF player1 = player2 THEN` ***[A][B]*** `  draw ← true` ***[C]*** `ENDIF` **Example 2** where the inverse is tested in a Boolean condition (italicised square brackets indicate where marks are awarded): `draw ← true` ***[C]*** `IF player1 ≠ player2 THEN` ***[A][B]*** `  draw ← false` ***[also needed for C]*** `ENDIF` **Example 3** (notes indicate where marks are awarded) 	**3**

10	**e**		Marks awarded as follows (allow any logically equivalent and correct answer). The marks are labelled **A** – **I** and shown in the examples where they are awarded:
			
			A. 1 mark for using selection that 'divides' the code for a draw from the code for when it is not a draw. This would probably be either two `IF` statements or an `IF-ELSE`;
			B. 1 mark for the correct Boolean condition(s) with the selection statements in mark **A**;
			C. 1 mark for outputting `'draw'`;
			D. 1 mark if the output from mark **C** is within the correct part of the selection statement;
			E. 1 mark for using selection with the correct condition(s) to ascertain which player won (**I.** if this and subsequent lines of code are not within the correct part of the selection from mark **A**);
			F. 1 mark for ensuring the winning player's choice will output first (even if the output is incorrect);
			G. 1 mark for outputting the player's choice (even if this is not the winning player);
			H. 1 mark for outputting the string `'beats'`;
			I. 1 mark for outputting the choice of the other player from that used in mark **F**;
			
			Example 1 (italicised square brackets indicate where marks are awarded):
			
			`IF draw = true THEN` ***[A] [B]***
			`  OUTPUT 'draw'` ***[C] [D]***
			`ELSE`
			`  IF player1HasWon = true THEN` ***[E]***
			`  OUTPUT options[player1]` ***[F] [G]***
			`  ELSE`
			`  OUTPUT options[player2]` ***[I]***
			`  ENDIF`
			`  OUTPUT "beats"` ***[H]***
			`  IF player1HasWon = true THEN`
			`  OUTPUT options[player2]` ***[also needed for F]***
			`  ELSE`
			`  OUTPUT options[player1]` ***[also needed for I]***
			`  ENDIF`
			`ENDIF`
				9

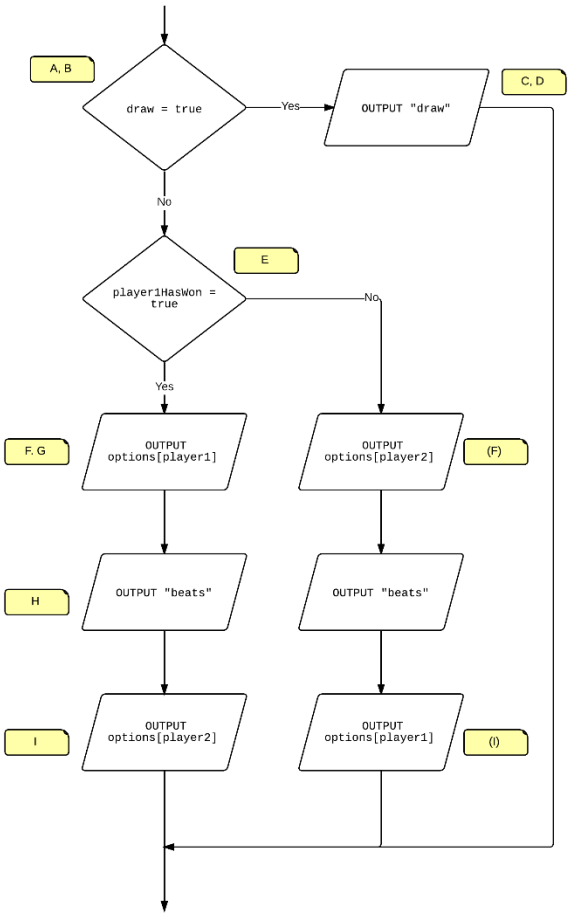
Example 2 (italicised square brackets indicate where marks are awarded):

```
IF draw = false THEN [A] [B]
  IF player1HasWon = true THEN [E]
    OUTPUT options[player1] [F] [G]
    OUTPUT "beats"  [H]
    OUTPUT options[player2] [I]
  ELSE
    OUTPUT options[player2] [also needed for F]
    OUTPUT "beats"  [also needed for H]
    OUTPUT options[player1] [also needed for I]
  ENDIF
ELSE
  OUTPUT "draw" [C] [D]
ENDIF
```

Example 3 (italicised square brackets indicate where marks are awarded):

```
IF player1HasWon = true THEN [A] [B] [E]
  OUTPUT options[player1] [F] [G]
  OUTPUT "beats" [H]
  OUTPUT options[player2] [I]
ENDIF
IF player1HasWon = false AND draw = false THEN
[also needed for A] [also needed for B] [also needed or E]
  OUTPUT options[player2] [also needed for F]
  OUTPUT "beats" [also needed for H]
  OUTPUT options[player1] [also needed for I]
ENDIF
IF draw = true THEN [also needed for A] [also needed
for B]
  OUTPUT "draw" [C] [D]
ENDIF
```

Example 4 (notes indicate where marks are awarded):


A, B
draw = true
Yes
OUTPUT "draw"
C, D
No
E
player1HasWon = true
No
Yes
F, G
OUTPUT options[player1]
OUTPUT options[player2]
(F)
H
OUTPUT "beats"
OUTPUT "beats"
I
OUTPUT options[player2]
OUTPUT options[player1]
(I)