AQA

AS LEVEL
Computing
COMP1
Mark scheme

2510
June 2015

Version/Stage: V1 Final Mark Scheme

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

Notation used in GCE Computing mark schemes:

; means a single mark
// means alternative response
/ means an alternative word or sub-phrase
A means acceptable creditworthy answer
R means reject answer as not creditworthy
I means ignore
NE means not enough

Qu	Part	Marking Guidance	Marks
1	01	0011 0111;	1
1	02	37;	1
1	03	Easier for people to read/understand; **R.** implication that it is easier for computers Can be displayed using fewer digits; More compact when printed/displayed; **NE.** Takes up less space **NE.** More compact **R.** Uses less storage space	MAX 1
1	04	1;1000101; **R.** if not 8 bits	2
1	05	101.10100 **R.** if not 8 bits **Mark as follows:** 3 bits before binary point correct; 5 bits after binary point correct; **Note for examiners** If the correct 8 bits are given (10110100) but with no binary point shown award 2 marks (only if all 8 bits are correct – if no binary point shown and any bit is incorrect then 0 marks) Award 1 mark if correct value represented but binary point in wrong place (e.g. 0101.1010)	2
1	06	011 0110; **R.** if not 7 bits	1
1	07	128 // 2^7;	1

<table>
<tr><td>1</td><td>08</td><td>Use the AND operator;
with the 7-bit ASCII code and the bit pattern 000 1111; A. 1001111 A. correct answers that use 8 bits instead of 7 bits A. denary/hexadecimal equivalents to the bit pattern (15 / F)
//
Use the XOR operator; A. EOR operator
with the 7-bit ASCII code and the bit pattern 0110000;
A. correct answers that use 8 bits instead of 7 bits A. denary/hexadecimal equivalents to the bit pattern (48 / 30)

Note for examiners
To get the 2nd mark point the bit pattern provided must work with the logical bitwise operator stated in the answer</td><td>2</td></tr>
<tr><td>1</td><td>09</td><td>0011 0000;</td><td>1</td></tr>
<tr><td>1</td><td>10</td><td>Recalculate the values for the parity bits using the data bits received (and compare these values with the parity bits received); A. check the parity bits
Add up the bit positions of the parity bits where a parity checks fails // add up the bit positions of the calculated parity bits that are different to those received;
The bit position of where the error has occurred is indicated; R. positions
The contents of the indicated bit position are flipped; R. positions</td><td>4</td></tr>
<tr><td>1</td><td>11</td><td>7;</td><td>1</td></tr>
<tr><td>1</td><td>12</td><td>No;</td><td>1</td></tr>
<tr><td>1</td><td>13</td><td><table><tr><th>Initial State</th><th>Input</th><th>New State</th></tr><tr><td>S_g</td><td>1</td><td>S_y;</td></tr><tr><td>S_y</td><td>0</td><td>S_y</td></tr><tr><td>S_y</td><td>1</td><td>S_y;</td></tr></table>A. 2nd and 3rd rows swapped

//
<table><tr><th>Initial State</th><th>Input</th><th>New State</th></tr><tr><td>S_g</td><td>1</td><td>S_y;</td></tr><tr><td>S_y</td><td>0 or 1</td><td>S_y;</td></tr><tr><td></td><td></td><td></td></tr></table></td><td>2</td></tr>
<tr><td>1</td><td>14</td><td>Works out if a given input is a (7-bit) ASCII code for a numeric character;</td><td>1</td></tr>
<tr><td>1</td><td>15</td><td>The arrow labelled with a 0 from state S_g should go to state S_j;</td><td>1</td></tr>
</table>

<table>
<tr><td>2</td><td>16</td><td>(Type of) shape // circle;
Coordinates of centre/midpoint;
Identifier;
(Length of) radius/diameter;
Line colour // outer colour;
Line width;
Fill colour // inner colour;

NE. Position/coordinates
NE. Colour
NE. Size
NE. Centre/midpoint</td><td>MAX 3</td></tr>
<tr><td>2</td><td>17</td><td>The image is divided into pixels;
Each possible colour is represented using a bit pattern // each pixel is represented using the same number of bits;
Information is stored about the colour of each pixel;
The position of the pixel in memory determines its location in the image;
A. metadata will be stored about the image</td><td>MAX 3</td></tr>
<tr><td>2</td><td>18</td><td>(For geometric images) less storage space /memory likely to be needed;
NE. less space
(For geometric images) will load faster (from secondary storage);
(For geometric images) will download faster;
Can be scaled/resized without distortion;
A. zoom
Image can be (more easily) searched for particular objects;
Can (more easily) manipulate individual objects in an image;
Can preserve the background so that it can be recreated if an object is deleted;</td><td>MAX 3</td></tr>
<tr><td>3</td><td>19</td><td><table><tr><th>IStr</th><th>OStr</th><th>Count</th></tr><tr><td>Lou</td><td></td><td>1</td></tr><tr><td></td><td>L</td><td>2</td></tr><tr><td></td><td>oL</td><td>3</td></tr><tr><td></td><td>uoL</td><td>4</td></tr></table>
Mark as follows:
1st column every cell left empty or contains the string Lou; NE. if no attempt to complete question
1st value in the 2nd column is L;
2nd value in the 2nd column is oL;
3rd value in the 2nd column is uoL;
3rd column correct;
I. speech marks around strings
I. case of letters</td><td>5</td></tr>
</table>

4	20	1. Correct variable declarations for `Prime`, `Count1` and `Count2`; **Note for examiners** If a language allows variables to be used without explicit declaration (eg Python) then this mark should be awarded if the three correct variables exist in the program code and the first value they are assigned is of the correct data type 2. Correct output message `The first few prime numbers are: ` ; 3. `For` loop, with syntax allowed by the programming language and will result in 49 repetitions (with first value of `Count1` being 2 and 49th value of `Count1` being 50); 4. `Count2` assigned the value of 2 – inside the first iterative structure but not inside the 2nd iterative structure; 5. `Prime` assigned the value of Yes – inside the first iterative structure but not inside the 2nd iterative structure; **I.** order of the statements assigning values to `Prime` and `Count2` 6. `While` loop, with syntax allowed by the programming language and correct condition for the termination of the loop; **A.** alternative correct logic for condition 7. 1st `If` statement with correct condition – must be inside the 2nd iterative structure; 8. `Prime` being assigned the value No inside the selection structure; 9. `Count2` incremented inside the 2nd iterative structure; **R.** if inside selection structure 10. 2nd `If` statement with correct condition – must be in the 1st iterative structure and not in the 2nd iterative structure; 11. Value of `Count1` being outputted inside the 2nd selection structure; **A.** Boolean data type for the variable `Prime` **I.** Case of variable names, strings and output messages **A.** Minor typos in variable names and output messages **I.** spacing in prompts **A.** initialisation of variables at declaration stage	11
4	21	****SCREEN CAPTURE**** *Must match code from 20, including messages on screen capture matching those in code. Code for 20 must be sensible.* **Mark as follows:** Correct printed output - 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47; **A.** any suitable format for printed list of numbers	1
4	22	Create a new variable called `Max`; **A.** any identifier for the variable **A.** no name specified for the variable Output a message (asking the user to enter a maximum value); Assign `Max` a value entered by the user; Change the condition for the 1st iteration structure so that it loops while `Count1` is less than `Max` (instead of less than or equal to 50);	MAX 3

<table>
<tr><td>5</td><td>23</td><td>BoardDimension;

R. if any additional code
R. if spelt incorrectly
I. case</td><td>1</td></tr>
<tr><td>5</td><td>24</td><td>DisplayWhoseTurnItIs // DisplayWinner // DisplayBoard // WriteWithMsg (VB6 only) // WriteLineWithMsg (VB6 only)// WriteLine (VB6 only)// WriteNoLine (VB6 only)// ReadLine (VB6 only);

R. if any additional code
R. if spelt incorrectly
I. case</td><td>1</td></tr>
<tr><td>5</td><td>25</td><td>Board;

R. if any additional code
R. if spelt incorrectly
I. case</td><td>1</td></tr>
<tr><td>5</td><td>26</td><td>Delete the three lines and add one copy of the line <u>after</u> the If statement(s);</td><td>1</td></tr>
<tr><td>5</td><td>27</td><td>If (PlayAgain contains) a lowercase letter; it is converted into uppercase;</td><td>2</td></tr>
<tr><td>5</td><td>28</td><td>The 123 in the 2nd condition should be 122; A. Change <= 123 to <123
The 3rd column should have condition values of N and N // the 1st column should have condition values of N and N;

There should only be an X in the last column; there should not be an X in any of the first three columns; //
there should be a Y (A. other sensible indicator) in the last column; there should be Xs (A. other sensible indicator) in the first three columns;

Note for examiners
Marks can be awarded for answers that show a corrected version of Table 4. An example of a possible correct Table 4:

<table>
<tr><td>Conditions</td><td>>=97</td><td>N</td><td>N</td><td>Y</td><td>Y</td></tr>
<tr><td></td><td><=122</td><td>N</td><td>Y</td><td>N</td><td>Y</td></tr>
<tr><td>Action</td><td>Change value of PlayAgain</td><td></td><td></td><td></td><td>X</td></tr>
</table>
</td><td>MAX 3</td></tr>
</table>

<table>
<tr>
<td>6</td>
<td>29</td>
<td>New variable <code>NoOfMoves</code> created with a numeric data type;
R. if spelt incorrectly I. case
Note for examiners
If a language allows variables to be used without explicit declaration (eg Python) then this mark should be awarded if a variable exists in the program code with the correct identifier and the first value it is assigned is of the correct data type

<code>NoOfMoves</code> initialised to zero at the start of a game;
A. different identifier if matches identifier for variable created
R. at declaration unless declaration is done at start of game (not start of program)

1 added to <code>NoOfMoves</code> after call to <code>MakeMove</code>;
A. different identifier if matches identifier for variable created
R. if adds 1 for an illegal move

Correct message <code>The number of moves completed so far:</code> displayed after call to <code>MakeMove</code> followed by a number;
I. Case of output message
A. Minor typos in output message
I. spacing in output message</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>****SCREEN CAPTURE****
Must match code from 29, including prompts on screen capture matching those in code. Code for 29 must be sensible.

<table>
<tr><td>1</td><td>BG</td><td>BE</td><td>BN</td><td>BM</td><td>BS</td><td>BN</td><td>BE</td><td>BG</td></tr>
<tr><td>2</td><td></td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td></tr>
<tr><td>3</td><td>BR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>WR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td>WG</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td></tr>
<tr><td>8</td><td></td><td>WE</td><td>WN</td><td>WM</td><td>WS</td><td>WN</td><td>WE</td><td>WG</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
</table>
Mark as follows:
Correct game position shown;
Correct message and value of 3 displayed;</td>
<td>2</td>
</tr>
</table>

7	31	`IF` statement with one correct condition; `IF` statement with a second correct condition; `IF` statement with all four correct conditions; Value of `False` returned to calling routine correctly if a square is out of bounds and value of `False` is still returned for all the original checks for illegal moves and value `True` of is still returned for all legal moves; **A.** two/four separate selection structures Note: the four conditions are `FinishRank > 8`, `FinishRank < 1`, `FinishFile < 1` and `FinishFile > 8` **A.** equivalent logic **Maximum of 3 marks** if the code, when run, would still execute the line `IF Board[FinishRank][FinishFile][1] ← "W" THEN` in the `CheckMoveIsLegal` subroutine when an out-of-bounds finish square is entered	4
7	32	****SCREEN CAPTURE**** *Must match code from 31, including prompts on screen capture matching those in code. Code for 31 must be sensible.* Finish square of 98 followed by message saying `That is not a legal move – please try again`; **R.** if the code, when run, would still execute the line `IF Board[FinishRank][FinishFile][1] ← "W" THEN` in the `CheckMoveIsLegal` subroutine when an out-of-bounds finish square is entered Finish square of 19 followed by message saying `That is not a legal move – please try again`; **R.** if the code, when run, would still execute the line `IF Board[FinishRank][FinishFile][1] ← "W" THEN` in the `CheckMoveIsLegal` subroutine when an out-of-bounds finish square is entered	3

<table>
<tr><td></td><td></td><td>Finish square of 86 followed by board position below being displayed;
A. value entered for finish square not displayed as long as correct board state is shown R. if no code in 31 that checks for <code>FinishFile</code> being valid

<table>
<tr><td>1</td><td>BG</td><td>BE</td><td>BN</td><td>BM</td><td>BS</td><td>BN</td><td>BE</td><td>BG</td></tr>
<tr><td>2</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td><td>BR</td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>WR</td></tr>
<tr><td>7</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td>WR</td><td></td></tr>
<tr><td>8</td><td>WG</td><td>WE</td><td>WN</td><td>WM</td><td>WS</td><td>WN</td><td>WE</td><td>WG</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
</table>
</td><td></td></tr>
<tr><td>8</td><td>33</td><td>option for <code>K</code> added to case/if statement with call to <code>CheckSarrumMoveIsLegal</code>; R. if any symbol other than <code>K</code> used

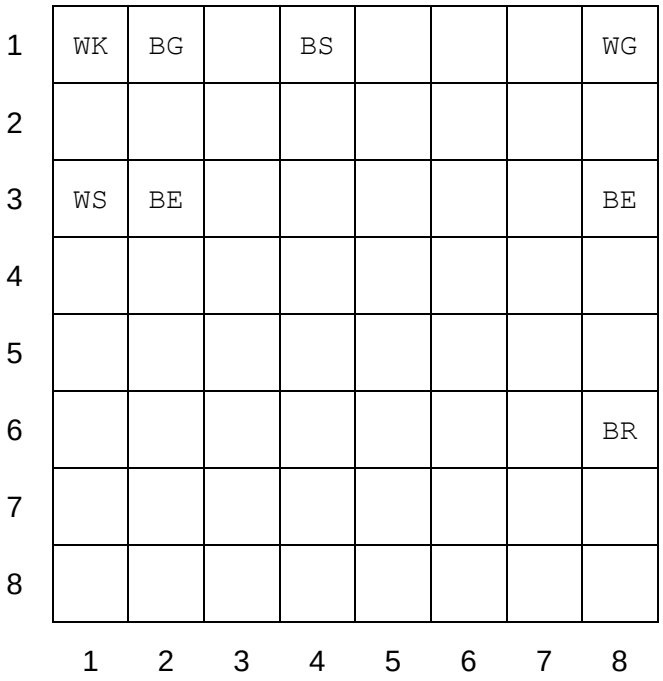
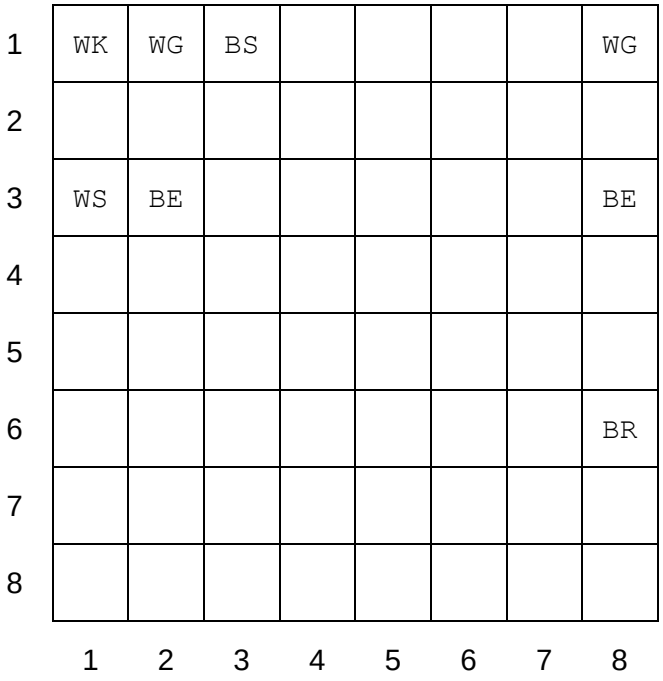
A. <code>CheckKashshaptuMoveIsLegal</code> (or similar) if evidence of creating new subroutine (or renaming existing subroutine) included somewhere in answer to question 8</td><td>1</td></tr>
<tr><td>8</td><td>34</td><td>1. White redum reaching 1st rank has symbol changed to <code>K</code> instead of <code>M</code>;

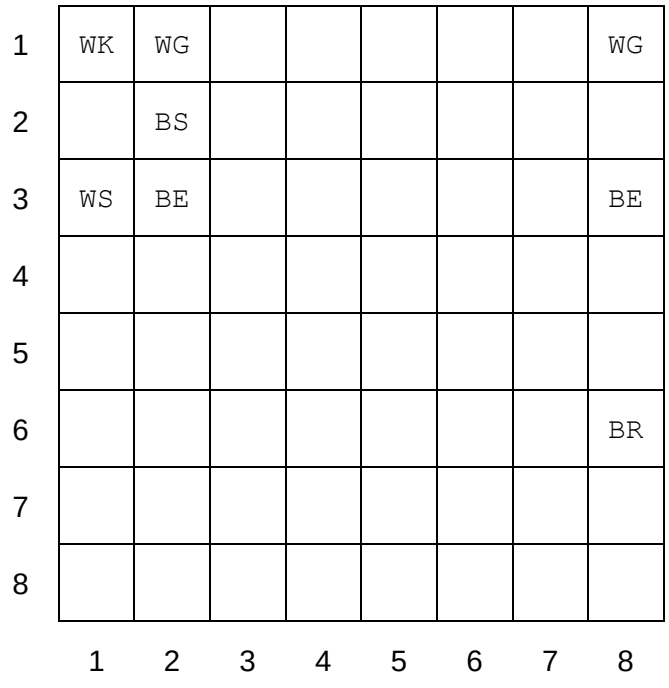
2. Added a selection structure with one of the following correct conditions:
• checks for the piece in the start square being a <code>K</code>
• checks for finish square not being empty // checks for finish square containing a black piece;

3. The additional selection structure has all necessary correct conditions and the correct logic;

4. Statement that changes the colour of a black piece in the finish square if it has been ‘captured’ by the kashshaptu - must be inside the selection structure;

5. When a kashshaptu moves and the finish square did not contain a black piece the contents of the start square become <code>" "</code> and if the finish square did contain a black piece the contents stay as <code>"WK"</code>;</td><td>5</td></tr>
</table>

8	35	****SCREEN CAPTURE**** *Must match code from 33 and 34, including prompts on screen capture matching those in code. Code for 33 and 34 must be sensible.* Finish square of 11 and board looks like diagram below;  3rd move finish square is 21 and board looks like the diagram below;  	3

		5th move has finish square of 22, board looks like diagram below and message `Black's sarrum has been captured. White wins!` is displayed;  	
9	36	1. New subroutine `GenerateFEN` created; **R.** if spelt incorrectly **I.** case 2. Correct parameters passed into the subroutine; 3. A string value will be returned by the subroutine; **R.** use of global variable 4. FEN record will have a `/` at end of each rank; 5. FEN record uses upper case for white pieces; 6. FEN record uses lower case for black pieces; 7. Each piece on the board in the FEN record (even if incorrectly represented eg WR instead of R); 8. Indicates where the empty spaces on the board are in the FEN record (even if incorrect representation); **A.** Incorrect counts of empty spaces 9. FEN Record correctly use `8` for an empty rank; **R.** is any character other than 8 in the FEN record for an empty rank 10. Correctly counts number of consecutive empty spaces // correctly works out the number of consecutive empty spaces;	13

		11. Count of empty spaces terminates at end of rank // calculation of number of empty spaces does not go over more than one rank; **R.** if no attempt to calculate/count number of empty spaces 12. FEN record shows whose move it is; 13. Accurate FEN record would be produced for every possible game state;	
9	37	Syntactically valid call to subroutine created in part 36; Value returned by subroutine is displayed; **R.** use of global variable Code for Task 2 added before the code asking the user to enter their move;	3
9	38	****SCREEN CAPTURE**** *Must match code from 36 and 37, including prompts on screen capture matching those in code. Code for 36 and 37 must be sensible.* **Mark as follows:** Sample game chosen and the FEN record returned/created by their subroutine from part 36 is displayed; Correct FEN record of `1g1s3G/R7/Se5e/8/8/7r/8/8/W` is displayed;	2

Pascal

Qu	Part	Marking Guidance	Marks

4	20	`Program Question4;`
		`  Var`
		`  Prime : String;`
		`  Count1 : Integer;`
		`  Count2 : Integer;`
		`  Begin`
		`  Writeln('The first few prime numbers are:')`
		`  For Count1 := 2 To 50`
		`  Do`
		`  Begin`
		`  Count2 := 2;`
		`  Prime := 'Yes';`
		`  While Count2 * Count2 <= Count1`
		`  Do`
		`  Begin`
		`  If (Count1 Mod Count2 = 0)`
		`  Then Prime := 'No';`
		`  Count2 := Count2 + 1`
		`  End;`
		`  If Prime = 'Yes'`
		`  Then WriteLn(Count1);`
		`  End;`
		`  ReadLn;`
		`  End.`
			11

6	29	`Repeat`
		`  NoOfMoves := 0;`
		`  WhoseTurn := 'W';`
		`  ...`
		`  Repeat`
		`  ...`
		`  Repeat`
		`  ...`
		`  Until MoveIsLegal;`
		`  MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn);`
		`  NoOfMoves := NoOfMoves + 1;`
		`  Writeln('The number of moves completed so far: ', NoOfMoves:3:1);`
		`  If GameOver`
		`  ...`
			4

7	31	`...`
		`Var`
		`  PieceType : Char;`
		`  PieceColour : String;`
		`  MoveIsLegal : Boolean;`
			4

		Begin
		  MoveIsLegal := True;
		  If (FinishFile = StartFile) And (FinishRank = StartRank)
		  Then MoveIsLegal := False
		  Else
		  If (FinishFile > 8) Or (FinishFile < 1) Or (FinishRank > 8) Or (FinishRank < 1)
		  Then MoveIsLegal := False
		  Else
		  Begin
		  PieceType := Board[StartRank, StartFile][2];
		  ...	

8	33	... 'S', **'K'** : MoveIsLegal := CheckSarrumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile); ...	1

8	34	...
		If (WhoseTurn = 'W') And (FinishRank = 1) And (Board[StartRank, StartFile][2] = 'R')
		  Then
		  Begin
		  Board[FinishRank, FinishFile] := 'W**K**';
		  Board[StartRank, StartFile] := ' ';
		  End
		  Else
		  Begin
		  If (Board[StartRank, StartFile][2] = 'K') And (Board[FinishRank, FinishFile] <> ' ')
		  Then Board[FinishRank, FinishFile] :=
		  Board[StartRank, StartFile][1] +
		  Board[FinishRank, FinishFile][2]
		  Else
		  If (WhoseTurn = 'B') And (FinishRank = 8) And (Board[StartRank, StartFile][2] = 'R')
		  Then
		  Begin
		  Board[FinishRank, FinishFile] := 'BM';
		  Board[StartRank, StartFile] := ' ';
		...
		
		Alternative answer
		...
		If (WhoseTurn = 'W') And (FinishRank = 1) And (Board[StartRank, StartFile][2] = 'R')
		  Then
		  Begin
		  Board[FinishRank, FinishFile] := 'W**K**';
		  Board[StartRank, StartFile] := ' ';
		  End
			5

```
  Else
    Begin
      If (Board[StartRank, StartFile][2] = 'K') And
(Board[FinishRank, FinishFile] <> '  ')
        Then Board[FinishRank, FinishFile] :=
             'W' + Board[FinishRank, FinishFile][2]
        Else
          If (WhoseTurn = 'B') And (FinishRank = 8) And
          (Board[StartRank, StartFile][2] = 'R')
            Then
              Begin
                Board[FinishRank, FinishFile] := 'BM';
                Board[StartRank, StartFile] := '  ';
...
```

9 36

```
Function GenerateFEN(Var Board : TBoard; WhoseTurn : Char)
: String;
  Var
    FEN : String;
    RankNo : Integer;
    FileNo : Integer;
    NoOfSpaces : Integer;
  Begin
    FEN := '';
    For RankNo := 1 To BoardDimension
      Do
        Begin
          NoOfSpaces := 0;
          For FileNo := 1 To BoardDimension
            Do
              Begin
                If Board[RankNo, FileNo] = '  '
                  Then NoOfSpaces := NoOfSpaces + 1
                  Else
                    Begin
                      If NoOfSpaces > 0
                        Then
                          Begin
                            FEN := FEN +
IntToStr(NoOfSpaces)Temp;
                            NoOfSpaces := 0;
                          End;
                      If Board[RankNo, FileNo][1] = 'B'
                        Then FEN := FEN +
Chr(Ord(Board[RankNo, FileNo][2]) + 32);
                        Else FEN := FEN + Board[RankNo,
FileNo][2]
                    End;
              End;
          If NoOfSpaces > 0
```

13

		Then FEN := FEN + IntToStr(NoOfSpaces);
		FEN := FEN + '/';
		End;
		FEN := FEN + WhoseTurn;
		GenerateFEN := FEN;
		End;
		
		Alternative answer – converting to lower case
		Else FEN := FEN + ansilowercase(Board[RankNo, FileNo][2]);
		
		Alternative answer – using Str instead of IntToStr
		Function GenerateFEN(Var Board : TBoard; WhoseTurn : Char)
		: String;
		Var
		...
		Temp : String;
		Begin
		...
		Str(NoOfSpaces, Temp);
		FEN := FEN + Temp;
		...	

9	37	... DisplayBoard(Board); **Writeln(GenerateFEN(Board, WhoseTurn));** DisplayWhoseTurnItIs(WhoseTurn); ...	3

VB.Net

Qu	Part	Marking Guidance	Marks

4	20	`Sub Main()`
		`  Dim Prime As String`
		`  Dim Count1 As Integer`
		`  Dim Count2 As Integer`
		`  Console.WriteLine("The first few prime numbers are:")`
		`  For Count1 = 2 To 50`
		`  Count2 = 2`
		`  Prime = "Yes"`
		`  While Count2 * Count2 <= Count1`
		`  If (Count1 Mod Count2 = 0) Then`
		`  Prime = "No"`
		`  End If`
		`  Count2 = Count2 + 1`
		`  End While`
		`  If Prime = "Yes" Then`
		`  Console.WriteLine(Count1)`
		`  End If`
		`  Next`
		`  Console.ReadLine()`
		`End Sub`
			11

6	29	`Do`
		`  NoOfMoves = 0`
		`  WhoseTurn = "W"`
		`  ...`
		`  Do`
		`  ...`
		`  Do`
		`  ...`
		`  Loop Until MoveIsLegal`
		`  GameOver = CheckIfGameWillBeWon(Board, FinishRank, FinishFile)`
		`  MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)`
		`  NoOfMoves = NoOfMoves + 1`
		`  Console.WriteLine("The number of moves completed so far: " & NoOfMoves)`
		`  If GameOver Then`
		`  ...`
			4

7	31	`...`
		`Dim PieceType As String`
		`Dim PieceColour As String`
		`If FinishFile = StartFile And FinishRank = StartRank Then`
		`  Return False`
		`End If`
		`If FinishFile > 8 Or FinishFile < 1 Or FinishRank > 8 Or`
			4

		FinishRank < 1 Then **Return False** **End If** PieceType = Board(StartRank, StartFile)(1) ...	
8	33	... Case "S", **"K"** Return CheckSarrumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile) ...	1

8	34	...
		If WhoseTurn = "W" And FinishRank = 1 And Board(StartRank, StartFile)(1) = "R" Then
		  Board(FinishRank, FinishFile) = "W**K**"
		  Board(StartRank, StartFile) = " "
		ElseIf Board(StartRank, StartFile)(1) = "K" And Board(FinishRank, FinishFile) <> " " Then
		  Board(FinishRank, FinishFile) = Board(StartRank, StartFile)(0) & Board(FinishRank, FinishFile)(1)
		ElseIf WhoseTurn = "B" And FinishRank = 8 And Board(StartRank, StartFile)(1) = "R" Then
		  Board(FinishRank, FinishFile) = "BM"
		  Board(StartRank, StartFile) = "
		Else
		  Board(FinishRank, FinishFile) = Board(StartRank, StartFile)
		  Board(StartRank, StartFile) = " "
		End If
		...
		
		Alternative answer
		...
		If WhoseTurn = "W" And FinishRank = 1 And Board(StartRank, StartFile)(1) = "R" Then
		  Board(FinishRank, FinishFile) = "W**K**"
		  Board(StartRank, StartFile) = " "
		ElseIf Board(StartRank, StartFile)(1) = "K" And Board(FinishRank, FinishFile) <> " " Then
		  Board(FinishRank, FinishFile) = "W" & Board(FinishRank, FinishFile)(1)
		ElseIf WhoseTurn = "B" And FinishRank = 8 And Board(StartRank, StartFile)(1) = "R" Then
		  Board(FinishRank, FinishFile) = "BM"
		  Board(StartRank, StartFile) = "
		Else
		  Board(FinishRank, FinishFile) = Board(StartRank, StartFile)
		  Board(StartRank, StartFile) = " "
		End If
		...
			5

9	36	Function GenerateFEN(ByVal Board(,) As String, ByVal WhoseTurn As Char) As String
		  Dim FEN As String
		  Dim RankNo As Integer
		  Dim FileNo As Integer
		  Dim NoOfSpaces As Integer
		  FEN = ""
		  For RankNo = 1 To BoardDimension
		  NoOfSpaces = 0
		  For FileNo = 1 To BoardDimension
		  If Board(RankNo, FileNo) = " " Then
		  NoOfSpaces = NoOfSpaces + 1
		  Else
		  If NoOfSpaces > 0 Then
		  FEN = FEN & CStr(NoOfSpaces)
		  NoOfSpaces = 0
		  End If
		  If Board(RankNo, FileNo)(0) = "B" Then
		  FEN = FEN & Board(RankNo, FileNo)(1).ToString.ToLower
		  Else
		  FEN = FEN & Board(RankNo, FileNo)(1)
		  End If
		  End If
		  Next
		  If NoOfSpaces > 0 Then
		  FEN = FEN & NoOfSpaces
		  End If
		  FEN = FEN & "/"
		  Next
		  FEN = FEN & WhoseTurn
		  Return FEN
		End Function
			13

9	37	... DisplayBoard(Board) **Console.WriteLine(GenerateFEN(Board, WhoseTurn))** DisplayWhoseTurnItIs(WhoseTurn) MoveIsLegal = False ...	3

VB6

Qu	Part	Marking Guidance	Marks

4	20	`Private Sub Form_Load()`
		`  Dim Prime As String`
		`  Dim Count1 As Integer`
		`  Dim Count2 As Integer`
		`  WriteLine ("The first few prime numbers are:")`
		`  For Count1 = 2 To 50`
		`  Count2 = 2`
		`  Prime = "Yes"`
		`  While Count2 * Count2 <= Count1`
		`  If (Count1 Mod Count2 = 0) Then`
		`  Prime = "No"`
		`  End If`
		`  Count2 = Count2 + 1`
		`  Wend`
		`  If Prime = "Yes" Then`
		`  WriteLine (Count1)`
		`  End If`
		`  Next`
		`End Sub`
		
		Alternative answers could use some of the following instead of WriteLine:
		`Console.Text = Console.Text & ...`
		`WriteLineWithMsg`
		`WriteWithMsg`
		`Msgbox`
		`WriteNoLine`
			11

6	29	`Do`
		`  **NoOfMoves = 0**`
		`  WhoseTurn = "W"`
		`  GameOver = False`
		`  ...`
		`  Do`
		`  ...`
		`  Do`
		`  ...`
		`  Loop Until MoveIsLegal`
		`  GameOver = CheckIfGameWillBeWon(Board, FinishRank, FinishFile)`
		`  Call MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)`
		`  **NoOfMoves = NoOfMoves + 1**`
		`  **WriteLine ("The number of moves completed so far: " & NoOfMoves)**`
		`  If GameOver Then`
		`  ...`
			4

7	31	...
		MoveIsLegal = True
		If FinishFile = StartFile And FinishRank = StartRank Then
		  MoveIsLegal = False
		Else
		  If FinishFile > 8 Or FinishFile < 1 Or FinishRank > 8 Or FinishRank < 1 Then
		  MoveIsLegal = False
		  Else
		  PieceType = Mid$(Board(StartRank, StartFile), 2, 1)
		  ...
			4

8	33	... Case "S", **"K"** MoveIsLegal = CheckSarrumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile) ...	1

8	34	...
		If WhoseTurn = "W" And FinishRank = 1 And Mid$(Board(StartRank, StartFile), 2, 1) = "R" Then
		  Board(FinishRank, FinishFile) = "W**K**"
		  Board(StartRank, StartFile) = " "
		ElseIf Mid$(Board(StartRank, StartFile), 2, 1) = "K" And Board(FinishRank, FinishFile) <> " " Then
		  Board(FinishRank, FinishFile) = Mid$(Board(StartRank, StartFile), 1, 1) & Mid$(Board(FinishRank, FinishFile), 2, 1)
		ElseIf WhoseTurn = "B" And FinishRank = 8 And Mid$(Board(StartRank, StartFile), 2, 1) = "R" Then
		  Board(FinishRank, FinishFile) = "BM"
		  Board(StartRank, StartFile) = " "
		Else
		  Board(FinishRank, FinishFile) = Board(StartRank, StartFile)
		  Board(StartRank, StartFile) = " "
		End If
		...
		
		Alternative answer
		...
		If WhoseTurn = "W" And FinishRank = 1 And Mid$(Board(StartRank, StartFile), 2, 1) = "R" Then
		  Board(FinishRank, FinishFile) = "W**K**"
		  Board(StartRank, StartFile) = " "
		ElseIf Mid$(Board(StartRank, StartFile), 2, 1) = "K" And Board(FinishRank, FinishFile) <> " " Then
		  Board(FinishRank, FinishFile) = "W" & Mid$(Board(FinishRank, FinishFile), 2, 1)
		ElseIf WhoseTurn = "B" And FinishRank = 8 And Mid$(Board(StartRank, StartFile), 2, 1) = "R" Then
			5

		Board(FinishRank, FinishFile) = "BM"
		Board(StartRank, StartFile) = " "
		Else
		Board(FinishRank, FinishFile) = Board(StartRank, StartFile)
		Board(StartRank, StartFile) = " "
		End If
		...	

9	36	Private Function GenerateFEN(ByRef Board() As String, ByVal WhoseTurn As String) As String
		Dim FEN As String
		Dim RankNo As Integer
		Dim FileNo As Integer
		Dim NoOfSpaces As Integer
		FEN = ""
		For RankNo = 1 To BoardDimension
		NoOfSpaces = 0
		For FileNo = 1 To BoardDimension
		If Board(RankNo, FileNo) = " " Then
		NoOfSpaces = NoOfSpaces + 1
		Else
		If NoOfSpaces > 0 Then
		FEN = FEN & CStr(NoOfSpaces)
		NoOfSpaces = 0
		End If
		If Mid$(Board(RankNo, FileNo), 1, 1) = "B" Then
		FEN = FEN & LCase(Mid$(Board(RankNo, FileNo), 2, 1))
		Else
		FEN = FEN & Mid$(Board(RankNo, FileNo), 2, 1)
		End If
		End If
		Next
		If NoOfSpaces > 0 Then
		FEN = FEN & NoOfSpaces
		End If
		FEN = FEN & "/"
		Next
		FEN = FEN & WhoseTurn
		GenerateFEN = FEN
		End Function
			13

9	37	... Call DisplayBoard(Board) **WriteLine (GenerateFEN(Board, WhoseTurn))** DisplayWhoseTurnItIs (WhoseTurn) MoveIsLegal = False ...	3

Java

Qu	Part	Marking Guidance	Marks

4	20	`static void main(string[] args) {`
		`  String prime;`
		`  int count1;`
		`  int count2;`
		
		`  console.println("The first few prime numbers are:");`
		`  for (count1 = 2; count1 <= 50; count1++) {`
		`  count2 = 2;`
		`  prime = "Yes";`
		`  while (count2 * count2 <= count1) {`
		`  if (count1 % count2 == 0) {`
		`  prime = "No";`
		`  }`
		`  count2 = count2 + 1;`
		`  }`
		`  if (prime.equals("Yes")) {`
		`  console.println(count1);`
		`  }`
		`  }`
		`  console.readln();`
		`}`
		
		Alternative solution :
		
		If not using `AQAConsole2015` class replace :
		`  console.println(. . .)`
		with
		`  System.out.println(. . .)`
			11

6	29	`do {`
		`  noOfMoves = 0;`
		`  whoseTurn = 'W';`
		`  ...`
		`  do {`
		`  ...`
		`  do {`
		`  ...`
		`  }`
		`  gameOver = checkIfGameWillBeWon(board, finishRank, finishFile);`
		`  makeMove(board, startRank, startFile, finishRank, finishFile, whoseTurn);`
		`  noOfMoves = noOfMoves + 1;`
		`  console.println("The number of moves completed so far: " + Float.tostring(noOfMoves));`
		`  if (gameOver) {`
		`  ...`
			4

7	31	...
		char pieceType;
		char pieceColour;
		boolean moveIsLegal = true;
		if ((finishFile == startFile) && (finishRank == startRank))
		{
		  moveIsLegal = false;
		}
		if (finishFile > 8 || finishFile < 1 || finishRank > 8 || finishRank < 1) {
		** moveIsLegal = false;**
		}
		pieceType = board[startRank][startFile].charAt(1);
		...
			4

8	33	... case 'S' : **case 'K' :** ...	1

8	34	...
		if ((whoseTurn == 'W') && (finishRank == 1) &&
		(board[startRank][startFile].charAt(1) == 'R')) {
		  board[finishRank][finishFile] = "W**K**";
		  board[startRank][startFile] = " ";
		} **else {**
		** if (board[startRank][startFile].charAt(1) == 'K'**
		&& !board[finishRank][finishFile].equals(" ")) {
		** Board[finishRank][finishFile] =**
		Character.toString(board[startRank][startFile].charAt(0)) +
		Character.toString(board[finishRank][finishFile].charAt(1))
		;
		** }** else {
		  if ((whoseTurn == 'B') && (finishRank == 8) &&
		(board[startRank][startFile].charAt(1) == 'R')) {
		  board[finishRank][finishFile] = "BM";
		  board[startRank][startFile] = " ";
		  } else {
		  board[finishRank][finishFile] =
		board[startRank][startFile];
		  board[StartRank][startFile] = " ";
		  }
		  }
		}
		...
		
		Alternative Solution
		...
		if ((whoseTurn == 'W') && (finishRank == 1) &&
			5

```
(board[startRank][startFile].charAt(1) == 'R')) {
  board[finishRank][finishFile] = "WK";
  board[startRank][startFile] = "  ";
} else {
  if (board[startRank][startFile].charAt(1) == 'K' &&
!board[finishRank][finishFile].equals("  ")) {
    board[finishRank][finishFile] = "W" +
Character.toString(board[finishRank][finishFile].charAt(1))
;
  } else {
    if ((whoseTurn == 'B') && (finishRank == 8) &&
(board[startRank][startFile].charAt(1) == 'R')) {
      board[finishRank][finishFile] = "BM";
      board[startRank][startFile] = "  ";
    } else {
      board[finishRank][finishFile] =
board[startRank][startFile];
      board[startRank][startFile] = "  ";
    }
  }
}
...
```

9 | 36

```
String generateFEN(String[][] board, char whoseTurn) {
  String FEN;
  int rankNo;
  int fileNo;
  int noOfSpaces;
  FEN = "";
  for (rankNo = 1; rankNo <= BOARD_DIMENSION; rankNo++) {
    noOfSpaces = 0;
    for (fileNo = 1; fileNo <= BOARD_DIMENSION; fileNo++) {
      if (board[rankNo][fileNo].equals("  ")) {
        noOfSpaces = noOfSpaces + 1;
      } else {
        if (noOfSpaces > 0) {
          FEN = FEN + Integer.toString(noOfSpaces);
          noOfSpaces = 0;
        }
        if (board[rankNo][fileNo].charAt(0) == 'B') {
          FEN = FEN + Character.toString(board[rankNo][
fileNo].charAt(1)).toLowerCase();
        } else {
          FEN = FEN + board[rankNo][fileNo].charAt(1);
        }
      }
    }
    if (noOfSpaces > 0) {
      FEN = FEN + Integer.toString(noOfSpaces);
    }
    FEN = FEN + "/";
```

13

		} FEN = FEN + Character.toString(whoseTurn); return FEN; }	

9	37	...
		moveIsLegal = false;
		displayBoard(board);
		c**console.println(generateFEN(board, whoseTurn));**
		displayWhoseTurnItIs(whoseTurn);
		...
			3

C#

Qu	Part	Marking Guidance	Marks

4	20	`static void Main(string[] args) {`
		`  string Prime;`
		`  int Count1;`
		`  int Count2;`
		`  Console.WriteLine("The first few prime numbers are:");`
		`  for (Count1 = 2; Count1 <= 50; Count1++) {`
		`  Count2 = 2;`
		`  Prime = "Yes";`
		`  while (Count2 * Count2 <= Count1) {`
		`  if (Count1 % Count2 == 0) {`
		`  Prime = "No";`
		`  }`
		`  Count2 = Count2 + 1;`
		`  }`
		`  if (Prime == "Yes") {`
		`  Console.WriteLine(Count1);`
		`  }`
		`  }`
		`  Console.ReadLine();`
		`}`
			11

6	29	`do`
		`{`
		`  **NoOfMoves = 0;**`
		`  WhoseTurn = 'W';`
		`  ...`
		`  do`
		`  ...`
		`  do`
		`  {`
		`  ...`
		`  } while (!MoveIsLegal)`
		`  GameOver = CheckIfGameWillBeWon(ref Board, FinishRank, FinishFile);`
		`  MakeMove(ref Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn);`
		`  **NoOfMoves = NoOfMoves + 1;**`
		`  **Console.WriteLine("The number of moves completed so far: " + NoOfMoves.ToString("f1"));**`
		`  if (GameOver)`
		` ...
			4

7	31	`...`
		`char PieceType;`
		`char PieceColour;`
		`Boolean MoveIsLegal = true;`
		`if ((FinishFile == StartFile) && (FinishRank == StartRank))`
			4

		MoveIsLegal = false; **if (FinishFile > 8 || FinishFile < 1 || FinishRank > 8 || FinishRank < 1)** **MoveIsLegal = false;** PieceType = Board[StartRank, StartFile][1]; ...	
8	33	... case 'S' : **case 'K' :** ...	1

8	34	...
		if ((WhoseTurn == 'W') && (FinishRank == 1) &&
		(Board[StartRank, StartFile][1] == 'R'))
		{
		Board[FinishRank, FinishFile] = "W**K**";
		Board[StartRank, StartFile] = " ";
		}
		else
		if (Board[StartRank, StartFile][1] == 'K' &&
		Board[FinishRank, FinishFile] != " ")
		Board[FinishRank, Finishfile] = Board[StartRank, StartFile][0].ToString + Board[FinishRank, Finishfile][1].ToString();
		else
		if ((WhoseTurn == 'B') && (FinishRank == 8) &&
		(Board[StartRank, StartFile][1] == 'R'))
		{
		Board[FinishRank, FinishFile] = "BM";
		Board[StartRank, StartFile] = " ";
		}
		else
		{
		Board[FinishRank, FinishFile] = Board[StartRank, StartFile];
		Board[StartRank, StartFile] = " ";
		}
		...
		
		Alternative Solution
		...
		if ((WhoseTurn == 'W') && (FinishRank == 1) && (Board[
		StartRank, StartFile][1] == 'R'))
		{
		Board[FinishRank, FinishFile] = "W**K**";
		Board[StartRank, StartFile] = " ";
		}
		else
		if (Board[StartRank, StartFile][1] == 'K' &&
			5

		Board[FinishRank, finishFile] != " ")
		Board[FinishRank, FinishFile] = "W" + Board[finishRank, FinishFile][1].ToString();
		else
		if ((WhoseTurn == 'B') && (FinishRank == 8) && (Board[StartRank, StartFile][1] == 'R'))
		{
		Board[FinishRank, FinishFile] = "BM";
		Board[StartRank, StartFile] = " ";
		}
		else
		{
		Board[FinishRank, FinishFile] = Board[StartRank, StartFile];
		Board[StartRank, StartFile] = " ";
		}
		...	

9	36	public static string GenerateFEN(string[,] Board, char WhoseTurn)
		{
		string FEN;
		int RankNo;
		int FileNo;
		int NoOfSpaces;
		FEN = "";
		for (RankNo = 1; RankNo <= BoardDimension; RankNo++)
		{
		NoOfSpaces = 0;
		for (FileNo = 1; FileNo <= BoardDimension; FileNo++)
		if (Board[RankNo, FileNo] == " ")
		NoOfSpaces = NoOfSpaces + 1;
		else
		if (NoOfSpaces > 0)
		{
		FEN = FEN + NoOfSpaces.ToString();
		NoOfSpaces = 0;
		}
		if (Board[RankNo, FileNo][0] == 'B')
		FEN = FEN + Board[RankNo, FileNo][1].ToString().ToLower();
		else
		FEN = FEN + Board[RankNo, FileNo][1];
		if (NoOfSpaces > 0)
		FEN = FEN + NoOfSpaces.ToString();
		FEN = FEN + "/";
		}
		FEN = FEN + WhoseTurn.ToString();
		return FEN;
		}
			13

9	37	... MoveIsLegal = false; DisplayBoard(ref Board); **Console.WriteLine(GenerateFEN(Board, WhoseTurn));** DisplayWhoseTurnItIs(WhoseTurn); ...	3

Python 2

Qu	Part	Marking Guidance	Marks

4	20	`print "The first few prime numbers are:"`
		`for Count1 in range(2,51):`
		`  Count2 = 2`
		`  Prime = "Yes"`
		`  while Count2 * Count2 <= Count1:`
		`  if Count1 % Count2 == 0:`
		`  Prime = "No"`
		`  Count2 = Count2 + 1`
		`  if Prime == "Yes":`
		`  print Count1`
			11

6	29	`while PlayAgain == "Y":`
		`  **NoOfMoves = 0**`
		`  WhoseTurn = "W"`
		`  ....`
		`  while not(GameOver):`
		`  ....`
		`  while not(MoveIsLegal):`
		`  ....`
		`  GameOver = CheckIfGameWillBeWon(Board, FinishRank, FinishFile)`
		`  MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)`
		`  **NoOfMoves = NoOfMoves + 1**`
		`  **print "The number of moves completed so far: ",NoOfMoves**`
		`  if GameOver:`
		`  DisplayWinner(WhoseTurn)`
		`  ....`
			4

7	31	`def CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):`
		`  MoveIsLegal = True`
		`  **if FinishFile > 8 or FinishFile < 1 or FinishRank > 8 or FinishFile < 1:**`
		`  **MoveIsLegal = False**`
		`  **elif** (FinishFile == StartFile) and (FinishRank == StartRank):`
		`  MoveIsLegal = False`
		`  ...`
			4

8	33	`if MoveIsLegal == True:`
		`  if PieceType == "R":`
		`  MoveIsLegal = CheckRedumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, PieceColour)`
		`  elif PieceType == "S" **or PieceType == "K"**:`
		` MoveIsLegal = CheckSarrumMoveIsLegal(Board, StartRank,
			1

		StartFile, FinishRank, FinishFile)
		  elif PieceType == "M":
		  ...	

8	34	if (WhoseTurn == "W") and (FinishRank == 1) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "W**K**"
		  Board[StartRank][StartFile] = " "
		elif Board[StartRank][StartFile][1] == "K" and
		Board[FinishRank][FinishFile] != " ":
		** Board[FinishRank][FinishFile] =**
		Board[StartRank][StartFile][0] +
		Board[FinishRank][FinishFile][1]
		elif (WhoseTurn == "B") and (FinishRank == 8) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "BM"
		  Board[StartRank][StartFile] = " "
		else:
		  Board[FinishRank][FinishFile] =
		Board[StartRank][StartFile]
		  Board[StartRank][StartFile] = " "
		  ...
		
		Alternative answer
		
		if (WhoseTurn == "W") and (FinishRank == 1) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "W**K**"
		  Board[StartRank][StartFile] = " "
		elif Board[StartRank][StartFile][1] == "K" and
		Board[FinishRank][FinishFile] != " ":
		** Board[FinishRank][FinishFile] = "W" +**
		Board[FinishRank][FinishFile][1]
		elif (WhoseTurn == "B") and (FinishRank == 8) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "BM"
		  Board[StartRank][StartFile] = " "
		else:
		  Board[FinishRank][FinishFile] =
		Board[StartRank][StartFile]
		  Board[StartRank][StartFile] = " "
		  ...
			5

9	36	def GenerateFEN(Board, WhoseTurn):
		  FEN = ""
		  for RankNo in range(1 , BOARDDIMENSION + 1):
		  NoOfSpaces = 0
		  for FileNo in range(1 , BOARDDIMENSION + 1):
		  if Board[RankNo][FileNo] == " ":
		  NoOfSpaces = NoOfSpaces + 1
			13

		else:
		if NoOfSpaces > 0:
		FEN = FEN + str(NoOfSpaces)
		NoOfSpaces = 0
		if Board[RankNo][FileNo][0] == "B":
		FEN = FEN + Board[RankNo][FileNo][1].lower()
		else:
		FEN = FEN + Board[RankNo][FileNo][1]
		if NoOfSpaces > 0:
		FEN = FEN + str(NoOfSpaces)
		FEN = FEN + "/"
		FEN = FEN + WhoseTurn
		return FEN	

9	37	while not(GameOver):
		DisplayBoard(Board)
		print GenerateFEN(Board, WhoseTurn)
		DisplayWhoseTurnItIs(WhoseTurn)
		MoveIsLegal = False
		while not(MoveIsLegal):
		...
			3

Python 3

Qu	Part	Marking Guidance	Marks

4	20	`print ("The first few prime numbers are:")`
		`for Count1 in range(2,51):`
		`  Count2 = 2`
		`  Prime = "Yes"`
		`  while Count2 * Count2 <= Count1:`
		`  if Count1 % Count2 == 0:`
		`  Prime = "No"`
		`  Count2 = Count2 + 1`
		`  if Prime == "Yes":`
		`  print (Count1)`
			11

6	29	`while PlayAgain == "Y":`
		`  **NoOfMoves = 0**`
		`  WhoseTurn = "W"`
		`  ...`
		`  while not(GameOver):`
		`  ...`
		`  while not(MoveIsLegal):`
		`  ...`
		`  GameOver = CheckIfGameWillBeWon(Board, FinishRank, FinishFile)`
		`  MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)`
		`  **NoOfMoves = NoOfMoves + 1**`
		`  **print ("The number of moves completed so far: "+str(NoOfMoves))**`
		`  if GameOver:`
		`  DisplayWinner(WhoseTurn)`
		`  ...`
			4

7	31	`def CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):`
		`  MoveIsLegal = True`
		`  **if FinishFile > 8 or FinishFile < 1 or FinishRank > 8 or FinishFile < 1:**`
		`  **MoveIsLegal = False**`
		`  **elif** (FinishFile == StartFile) and (FinishRank == StartRank):`
		`  MoveIsLegal = False`
		`  ...`
			4

8	33	`if MoveIsLegal == True:`
		`  if PieceType == "R":`
		`  MoveIsLegal = CheckRedumMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, PieceColour)`
		`  elif PieceType == "S" **or PieceType == "K"**:`
		` MoveIsLegal = CheckSarrumMoveIsLegal(Board,
			1

		StartRank, StartFile, FinishRank, FinishFile)
		  elif PieceType == "M":
		  ...	

8	34	if (WhoseTurn == "W") and (FinishRank == 1) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "W**K**"
		  Board[StartRank][StartFile] = " "
		**elif Board[StartRank][StartFile][1] == "K" and
		Board[FinishRank][FinishFile] != " ":
		  Board[FinishRank][FinishFile] =
		Board[StartRank][StartFile][0] +
		Board[FinishRank][FinishFile][1]**
		elif (WhoseTurn == "B") and (FinishRank == 8) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "BM"
		  Board[StartRank][StartFile] = " "
		else:
		  Board[FinishRank][FinishFile] =
		Board[StartRank][StartFile]
		  Board[StartRank][StartFile] = " "
		  ...
		
		Alternative answer
		
		if (WhoseTurn == "W") and (FinishRank == 1) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "W**K**"
		  Board[StartRank][StartFile] = " "
		**elif Board[StartRank][StartFile][1] == "K" and
		Board[FinishRank][FinishFile] != " ":
		  Board[FinishRank][FinishFile] = "W" +
		Board[FinishRank][FinishFile][1]**
		elif (WhoseTurn == "B") and (FinishRank == 8) and
		(Board[StartRank][StartFile][1] == "R"):
		  Board[FinishRank][FinishFile] = "BM"
		  Board[StartRank][StartFile] = " "
		else:
		  Board[FinishRank][FinishFile] =
		Board[StartRank][StartFile]
		  Board[StartRank][StartFile] = " "
		  ...
			5

9	36	def GenerateFEN(Board, WhoseTurn):
		  FEN = ""
		  for RankNo in range(1 , BOARDDIMENSION + 1):
		  NoOfSpaces = 0
		  for FileNo in range(1 , BOARDDIMENSION + 1):
		  if Board[RankNo][FileNo] == " ":
		  NoOfSpaces = NoOfSpaces + 1
		  else:
			13

		`        if NoOfSpaces > 0:`
		`          FEN = FEN + str(NoOfSpaces)`
		`          NoOfSpaces = 0`
		`        if Board[RankNo][FileNo][0] == "B":`
		`          FEN = FEN + Board[RankNo][FileNo][1].lower()`
		`        else:`
		`          FEN = FEN + Board[RankNo][FileNo][1]`
		`    if NoOfSpaces > 0:`
		`        FEN = FEN + str(NoOfSpaces)`
		`    FEN = FEN + "/"`
		`  FEN = FEN + WhoseTurn`
		`  return FEN`	

9	37	`while not(GameOver):`
		`  DisplayBoard(Board)`
		`  print (GenerateFEN(Board, WhoseTurn))`
		`  DisplayWhoseTurnItIs(WhoseTurn)`
		`  MoveIsLegal = False`
		`  while not(MoveIsLegal):`
		`  ...`
			3