4.2 Regular languages mark scheme

Q1.

(a) **Mark is for AO2 (analyse)**

Input string is a (valid) postcode followed by additional characters // the input string is not a valid (UK) postcode // the mail will not be put in any of the three vans;

NE. The input string is not a valid IP postcode
A. Postcode has additional characters at the end
A. Postcode is too long

1

(b) **Mark is for AO2 (analyse)**

(The string represents) an IP postcode that is not for a location in the town of Ipswich //
(The string represents) an IP postcode that is for a location near Ipswich //
(The string represents) a postcode for a letter that needs to go in Van B;

NE. valid postcode

1

(c) **Mark is for AO2 (analyse)**

(IP / two letters) followed by number, letter, (number, letter, letter) //
(IP / two letters) followed by number between 5 and 9, number, (number, letter, letter) //
IP followed by 0;

A. postcodes that only have one letter at the start

1

EXAM PAPERS PRACTICE

(d) **Marks are for AO2 (apply)**

```
\a?\a;\d;(\a|\d)?;\d\a\a; //
\a\a?;\d;(\a|\d)?;\d\a\a; //
\a?\a;\d;(\d|\a)?;\d\a\a; //
\a\a?;\d;(\d|\a)?;\d\a\a;
```

Mark as follows:

1 mark:
1. regular expression can start with either one or two letters **R**. if more than two letters allowed

1 mark:
2. regular expression has a numeric digit after the initial letters **A**. if more than the correct number of letters allowed
//
regular expression has a numeric digit before it allows a single, optional letter or numeric digit

1 mark:
3. regular expression allows a single, optional letter or numeric digit after the first numeric digit in the expression

//
regular expression allows a single, optional letter or numeric digit before the numeric digit followed by exactly two letters at the end of the expression

1 mark:
4. regular expression ends with a numeric digit followed by exactly two letters

MAX 3 if final answer is not correct

R. any mark points after 2nd use of | metacharacter
A. suitable alternatives to `\a` and `\d` e.g. use of `[A-Z]`, `[a-z]` or `[A-Za-z]` instead of `\a` and `[0-9]` instead of `\d`

DPT. / instead of \

4

[7]

Q2.

(a)

Original State	Input	New State
S30	10	S40
S30	20	S50
S30	50	S0
S30	R	S0

Note: order of rows not important

Mark as follows:
Any 2 rows correct;
All 3 rows correct;

2

(b) 5;

1

[3]

EXAM PAPERS PRACTICE

Q3.

(a) Parity Bit: 1;

Start bit, Stop Bit: Can be either 0 or 1, but must both be different to get mark;

2

(b) **Definition (1 mark):**

Receiver and transmitter (clocks) do not need to be/are not (exactly) synchronised // transmission of data without use of external clock signal // receiver and transmitter clock only synchronised at start of/for length of transmission // data sent as soon as available rather than waiting for clock pulse/ synchronisation symbol;

Explanation of start and stop bits (max 2 marks):

Start bit synchronises receiver (clock) (to transmitter/data) // locks receiver and transmitter in phase // starts receiver's

clock // wakes receiver;
Stop bit allows start bit to be recognised // allows receiver to process received bits;
A. Start and stop bits indicate when data is being transmitted/ begins – if neither of the other two marks for start and stop bits have been awarded

3

(c) 1010001;

A. Separator between digits e.g. comma

1

(d) It is the parity bit;
A. Odd parity bit
A. If there are an even or odd number of 1s in the input

1

(e) Only a small quantity of data to send // data transmission speed not important;
Widespread availability of USB/serial connections;
Serial communication avoids crosstalk // interference between signals on each wire;
Serial communication avoids data skew;
A. Serial communication is cheaper to implement with a suitable reason given
A. For future flexibility if devices were moved further apart
N.E. Serial is less error prone / fewer errors

MAX 2

2

[9]

Q4.

(a) Syntax diagram;
A. Railroad diagram

1

(b) 3

1

(c) **Mark is conditional upon a correct answer to (b)**
It requires that a signed binary number starts with a + or a - // it won't allow a signed binary number that starts with a bit/digit/1 or 0 // + and – are not optional /are required;
A. 'String' or 'number' for 'signed binary number'

1

(d) Some example correct regular expressions are listed below but award a mark for any regular expression that would correctly represent the language accepted by the FSA.

a ((ba)|c)* // a((ba)*|c*)* // a((ba)*|c)* // a((ba)|c*)* // a((ba)*|c*)+ // a(c*(ba)?)

A. Missing brackets around ba as BOD

1

[4]

EXAM PAPERS PRACTICE

Q5.

(a) **All marks AO3 (programming)**

Python 2.6:

```
print "How far to count?"
HowFar = input()
While HowFar < 1:
    print "Not a valid number, please try again."
    HowFar = input()
for MyLoop in range(1,HowFar+1):
    if MyLoop%3 == 0 and MyLoop%5 == 0:
        print "FizzBuzz"
    elif MyLoop%3 == 0:
        print "Fizz"
    elif MyLoop%5 == 0:
        print "Buzz"
    else:
        print MyLoop
```

1 mark: Correct prompt `"How far to count?"` followed by `HowFar` assigned value entered by user;
1 mark: `WHILE` loop has syntax allowed by the programming language and correct condition for the termination of the loop;
1 mark: Correct prompt `"Not a valid number, please try again."` followed by `HowFar` being assigned value entered by the user (must be inside iteration structure);
1 mark: Correct syntax for the `FOR` loop using correct range appropriate to language;
1 mark: Correct syntax for an `IF` statement, including a `THEN` and `ELSE / ELIF` part;
1 mark: Correct syntax for `MyLoop MOD 5 = 0` and `MyLoop MOD 3 = 0` used in the `IF` statement(s);
1 mark: Correct output for cases in the selection structure where `MyLoop MOD 3 = 0` or `MyLoop MOD 5 = 0` or both - outputs `"FizzBuzz"`, `"Fizz"` or `"Buzz"` correctly;
1 mark: Correct output (in the `ELSE` part of selection structure), when `MyLoop MOD 3  0` and `MyLoop MOD 5  0` – outputs value of `MyLoop`;

8

EXAM PAPERS PRACTICE

(b) **All marks AO3 (evaluate)**

Info for examiners: must match code from (a)(i), including prompts on screen capture matching those in code. Code for (a)(i) must be sensible.

First Test

```
How far to count?
18
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
```

```
11
Fizz
13
14
FizzBuzz
16
17
Fizz
```

Second Test

```
How far to count?
-1
Not a valid number, please try again.
```

Screenshot with user input of `18` and correct output and user input of -1 and correct output;

A different formatting of output eg line breaks

1

(c) **Mark is for AO2 (analysis)**

A `FOR` loop is used as it is to be repeated a known number of times;

1

(d) **All marks AO2 (analysis)**

Example of input:
[nothing input]
[a string] for example: `12A`

Method to prevent:
can protect against by using a try,except structure / / exception handling;
test the input to see if digits only;
convert string to integer and capture any exception;
use a repeat / while structure / / alter repeat / while to ask again until valid data input;

EXAM PAPERS PRACTICE

1 mark: Example of input
Max 2 marks: Description of how this can be protected against

3

(e) **All marks AO1 (understanding)**

Use of indentation to separate out statement blocks;
Use of comments to annotate the program code;
Use of procedures / functions / sub-routines;
Use of constants;
Max 3, any from 4 above

3

(f) **All marks AO2 (apply)**

Input string	Accepted by FSM?
`aaab`	YES

abbab	NO
bbbbba	YES

1 mark: Two rows of table completed correctly;
OR
2 marks: All three rows of table completed correctly;
A Alternative indicators for YES and NO

2

(g) **All marks AO2 (apply)**
1 mark: a string containing zero or more (**A** 'any number of') b characters;
1 mark: and an odd amount of a characters;
N.E. all strings containing an odd number of characters

2

[20]

Q6.

(a) **Mark is for AO1 (understanding)**

Original state	Input	New state
S3	0	S4
S3	1	S2

1 mark: Table completed as above
I order of rows

1

(b) **All marks AO2 (analyse)**

(0|1)*((00)|(11))(0|1)*

EXAM PAPERS PRACTICE

Mark as follows:
1 mark: (0|1)* at start;
1 mark: (00)|(11);
1 mark: (0|1)* at end;

Or

Alternative answer
(0|1)*(11(0|1)*)|(00(0|1)*)

Mark as follows:
1 mark: (0|1)* at start;
1 mark: (11(0|1)*);
1 mark: |(00(0|1)*) at end;

Maximum 2 marks: If final answer not correct.

A any regular expression that correctly defines the language.

3

(c) **Mark is for AO2 (apply)**

Rule number (given in Figure 2)	Could be defined using a regular expression
1	Y
2	Y
3	Y
4	N
5	N
6	Y

1 mark: All values in the table have been completed correctly.

1

(d) **1 mark for AO2 (analyse) and 1 mark for AO3 (design)**

1 mark for AO2 (analyse): There is no non-recursive / base case;
1 mark for AO3 (design): `<word> ::= <char><word> | <char>;`

2

[7]

Q7.

(a) 0011 0111;

1

(b) 37;

1

EXAM PAPERS PRACTICE

(c) 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

(d) 1;1000101;
R if not 8 bits

2

(e) 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

(f) 011 0110;

R if not 7 bits

1

(g) 128 //
2^7;

1

(h) 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

2

(i) 0011 0000;

1

(j) 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

4

(k) 7;

1

(l) No;

1

(m)

Initial State	Input	New State
S_g	1	S_y;

EXAM PAPERS PRACTICE

S_y	0	S_y
S_y	1	S_y;

A 2nd and 3rd rows swapped

//

Initial State	Input	New State
S_g	1	S_y;
S_y	0 or 1	S_y;

2

(n) Works out if a given input is a (7-bit) ASCII code for a numeric character;

1

(o) The arrow labelled with a 0 from state S_g should go to state S_j;

1

[22]

Q8.

(a)

Current State	S_3	S_3	S_3
Input Symbol	a	b	c
Next State	S_6	S_6	S_4

EXAM PAPERS PRACTICE

1 mark for all six correct values in the bold rectangular area
The columns do not have to be in the same order as shown, but the pairings must be correct i.e. (a - S_6, b - S_6, c - S_4).
A 4 for S_4 and 6 for S_6

1

(b) S_3
A 3
I An additional name given to the state eg “State 3”

1

(c) To ensure that a non-valid string is trapped // prevent the accepting state being reached;
A To capture invalid input
A To capture strings that are too long / have extra characters
NE Infinite loop / state cannot be left

1

(d) `a(bc)*a // a(bc)+?a // (aa)|(a(bc)*a) // (aa)|(a(bc)+?a) // (aa)|(a(bc)+a)`

1 mark for recognising an a at both ends
1 mark for correctly recognising 0 or more repetitions of bc
I ^ and $ at start and end of expression
A Any type of bracket

2

(e) Turing Machine has (an infinite / unlimited amount of) memory / storage;
Turing Machine can read and write / input and output (data) to / from a tape;
Turing Machine has infinitely long tape;
NE Turing Machine has a tape
MAX 1

1

[6]

Q9.

(a) (i) Zero or more `b`s followed by a / one `c`;
A answers by example but must be at least `c, bc, bbc` and indicate the sequence continues.

1

(ii) Zero or one `b`s followed by (a / one) `c` / / the strings `c` or `bc`;

1

(b) Correct expression: `b(cd)*(e|fg)`
A use of incorrect bracket types
A accept brackets around `fg`
A `(cd)+?` for `(cd)*`
I ^ at start, $ at end of expression
2 marks for the full correct expression.
1 mark for including either `(cd)*` or `(e|fg)` in an incorrect expression.

2

[4]

Q10.

(a) 167;;

EXAM PAPERS PRACTICE

If final answer is incorrect **Max 1** can be awarded for some correct working out being shown by the candidate:

1010 0111;
10 * 16 // 160 // A * 16;
A = 10;
Multiplying a value by 16 and adding on 7;

2

(b) 0111.1010 // 01111010

Mark as follows:
4 bits before binary point are 0111;
4 bits after binary point are 1010;

2

(c) 1;110 1110;
R if not 8 bits

2

(d) 127;

1

(e) The number to subtract is converted into a negative number;
NE Convert into two□s complement
This is then added to the first number;

Two marks for example:

23 = 00010111
-48 = 11010000;

11100111; (= -25)

A if not used 8 bits in examples
A 23 + - 48 is worth 1 mark only (if there is no description)

Note: for the first mark in the example to be awarded the two bit patterns must be correct. For the second mark in the example accept an incorrect answer as long as it is a correct addition using one of the two correct bit patterns.

4

(f) 11101110;
R 01110111

1

(g) 11101011;
DPT A. 11010111

1

(h) Get the two's complement (of a positive binary value) //
Converts a positive binary value into its negative equivalent;
A It inverts all bits after the first 1 is received;

1

(i)

Input	Original State	Output	New State
0	S0	0	S0
1	S0	1	S1
0	S1	1	S1
1	S1	0	S1

Mark as follows:
S0 as original state for 2nd row;
1 as output for 3rd row;
Final row correct;

3

[17]

Q11.

(a) (i) ❷ S_1 **A** 1, State 1
❸ S_T **A** T, State T

Both answers correct to get mark;

1

(ii) $\delta (S_B, 0) = (S_0, x, \longrightarrow)$;

A 0, x, ⟶ or 0 | x | ⟶;
R if additional rules listed
I minor transcription errors e.g. missing , (δ

1

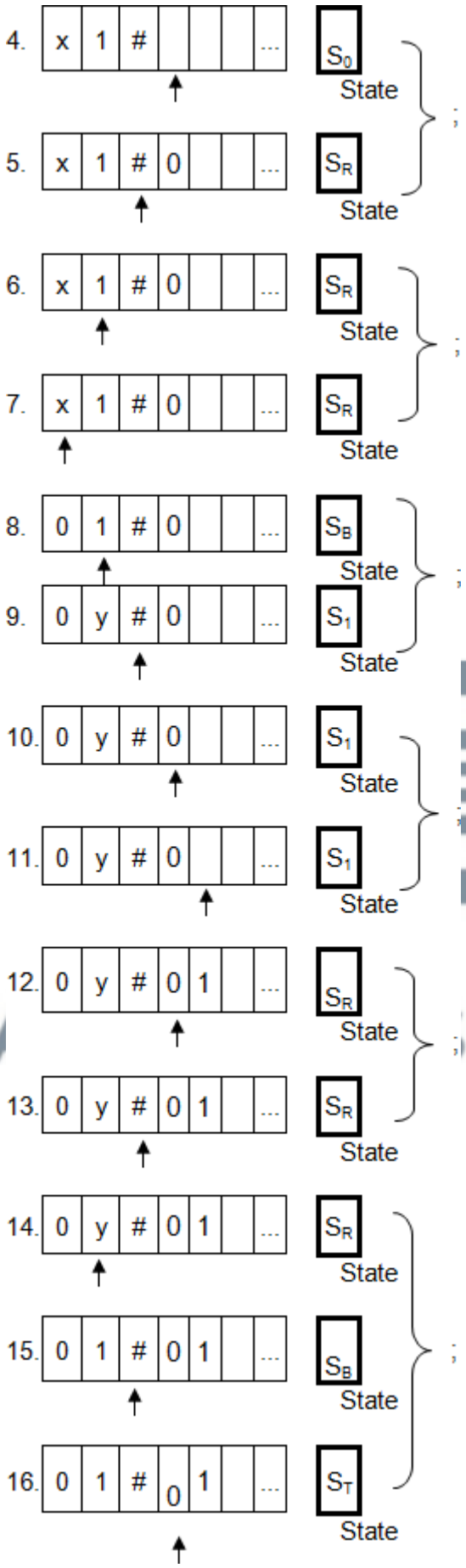
(iii) $\delta (S_R, x) = (S_B, 0, \longrightarrow)$ and $\delta (S_R, y) = (S_B, 1, \longrightarrow)$;

A x, 0, ⟶ or x | 0 | ⟶ and y, 1, ⟶ or y | 1 | ⟶
R if additional rules listed
I minor transcription errors e.g. missing , (δ

1

(b) One mark per bracketed section.

EXAM PAPERS PRACTICE



Must have correct tape contents and state for each mark
A blank symbols instead of empty cells

EX PRACTICE

DPT If the read / write head is not drawn on some rows, this should result in the loss of the mark on the first occasion that it is missing only.
Marks should be awarded for subsequent rows, even if the read / write head is not drawn.

6

(c) (i) Mark symbol currently being copied // to indicate how much of the string has been copied so far // to indicate where to return to (to copy next symbol);
A placeholders
NE x represents 0, y represents 1

1

(ii) Copy a string // copy a binary number // copy a bit pattern;
A Repeat

1

[11]

Q12.

(a)

Original State	Input	New State	
S0	10	S10	
S0	20	S20	;
S0	50	S50	;
S0	R	S0	;

Note: order of completed rows not important

3

(b) 20, 20, 10;
R, R, 50;
10, 20, 20;
20, 50, 50;
20, R, 50;

Max 4

[7]

EXAM PAPERS PRACTICE

Q13.

(a) Backus-Naur (Form);
A Backus Normal (Form), BNF, Extended Backus-Naur (Form), Augmented Backus-Naur (Form), ABNF
A Misspellings of Backus-Naur
A Format for Form and the word “Notation”
R BN

1

(b)

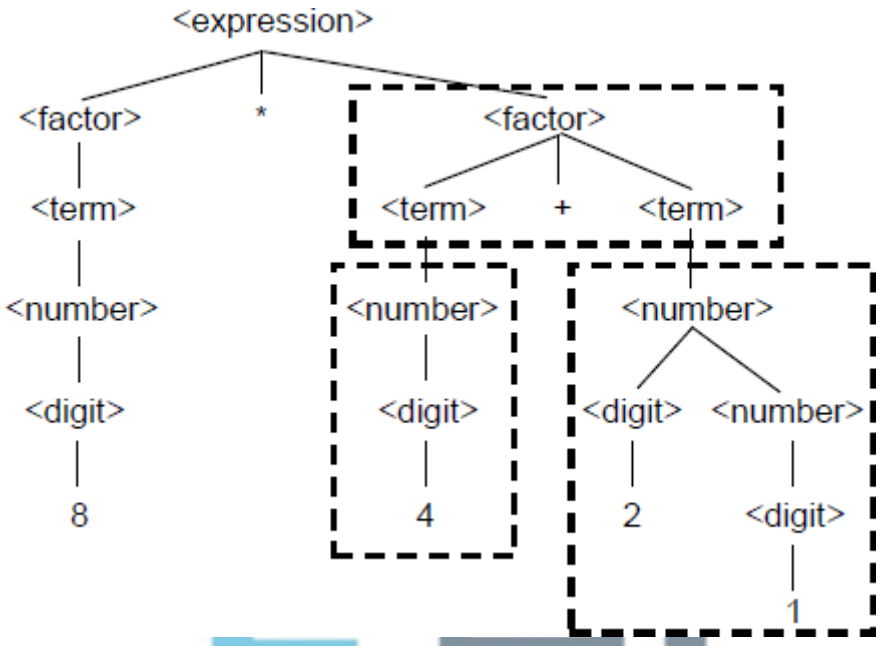
Statement Type	String	Valid (Yes / No)
<number>	129.376	*No;*

<factor>	23 + 17	*Yes;*

A Alternative clear indicators of Yes or No e.g. Y / N, Tick / Cross, Valid / Invalid, True / False

2

(c)



1 mark for each area surrounded by a rectangle

A missing chevrons
DPT Arrows drawn instead of lines

3

[6]

EXAM PAPERS PRACTICE

Q14.

(a) Greater the bandwidth, the higher the bit rate // positive correlation // (directly) proportional;
Bandwidth must be at least 2wHz where w is the bit rate in bits per second;

Max 1

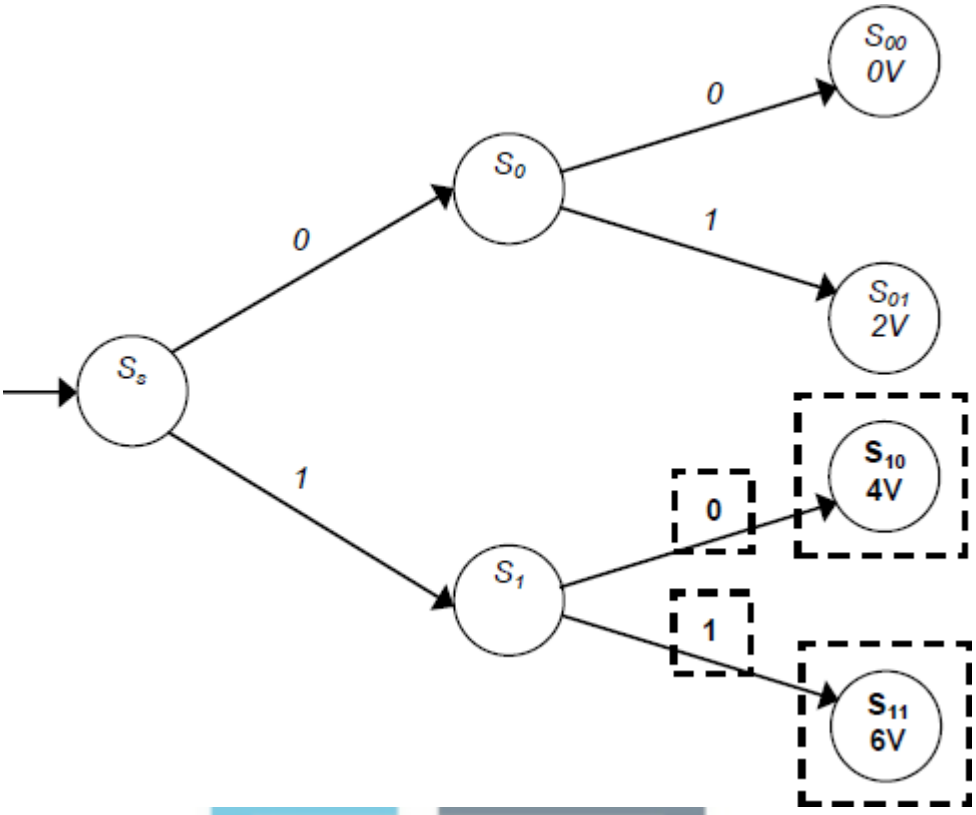
(b) Time delay between the moment something is initiated and the moment its effect begins
A time delay between signal being transmitted and arriving
A time taken for transmitted data to arrive at the receiver
A lag for time delay
NE delay in transmission, transmission time

1

(c) Bit rate is double / twice baud rate // Baud rate is half bit rate;
A "It" is double;
A 2:1

1

(d)



1 mark for labelling a transition arrow with 0
1 mark for labelling a transition arrow with 1
1 mark for labelling a state with the value 4V and a unique state name
1 mark for labelling a state with the value 6V and a unique state name

Max 2 if the states and transition arrow labels do not correspond

Note that:

- The state names do not have to match those given here.
- The voltage values can be followed by a V, the word Volts or nothing.
- The zero and one on the transition arrows to the right of S_1 can be either way around e.g. 1 above 0 is okay.

4

[7]

EXAM PAPERS PRACTICE

Q15.

(a) Is it possible in general to ***write a program / algorithm;*** that can tell, given any
program and its inputs and without ***running / executing the program;***, whether
the given program with its given inputs will halt?
A "it" in second reference to program.
A "create a Turing machine" for "write an algorithm"

2

(b) Shows that some problems are non-computable / undecideable // shows that some problems cannot be solved by a computer / algorithm;

In general, inspection alone cannot always determine whether any given algorithm will halt for its given inputs // a program cannot be written that can

determine whether any given algorithm will halt for its given inputs;
A it is not computable

Max 1

[3]

Q16.

(a) ab $^{+}$ c // abb * c // ab * bc;
I ^ at start, $ at end of expression

1

(b) (0|1)1* // (1|0)1* // [01]1*// [10]1* // [0|1]1* // [1|0]1* // 0 | (0?1^{+})
I ^ at start, $ at end of expression

1

[2]

Q17.

(a)

Current State	S_1	S_1	S_2	S_2	S_3	S_3	S_4	S_4	S_5	S_5
Input Symbol	0	1	0	1	0	1	***0***	***1***	***0***	***1***
Next State	S_2	S_3	S_2	S_4	S_3	S_3	***S_4***	***S_5***	***S_5***	***S_4***

1 mark for all four bolded columns correct
A the two columns for S_4 either way round and similar for S_5

1

(b) Accept/Accepting/Accepted (state) // Input (string) is accepted
A if the FSA finishes in this state output is Yes
R Stop state

1

(c) **Input String**	**String Accepted? (Yes / No)**
101	***No***
000	***No***
010001101	***No***
0100011011	***Yes***

1 mark for any two correct answers
2 marks for all four answers correct

2

(d) Strings that start with a 0;
A does not start with 1
R starts with 00, 01, any statement of a specific second digit being required

Followed by any sequence containing an odd number of 1s and zero or more 0s;
A String with an odd number of 1s in it.

EXAM PAPERS PRACTICE

A Numbers or bit patterns in place of 0s and 1s.

2

[6]

Q18.

(a) (i) One or more a's followed by (a/one) b;
A answers by example but must be at least ab, aab, aaab and show that the sequence continues.

1

(ii) The strings ab or b // zero or one a's followed by (a / one) b

1

(iii) A sequence of 0 or more occurrences of ab.
A answers by example but must be at least the empty string, ab, abab, ababab and show that the sequence continues.

1

(b) (i) Clai?re // Clare|Claire
A other valid possibilities e.g. Cla(ir|r)e, Cl(air|ar)e
A use of different types of brackets

1

(ii) 10(0|1)*01

1 mark for the 10 at the start and 01 at the end
1 mark for (0|1) in the middle to produce a correct expression*

A use of different types of brackets
Award 2 marks for any other expression that would work

2

[6]

Q19.

EXAM PAPERS PRACTICE

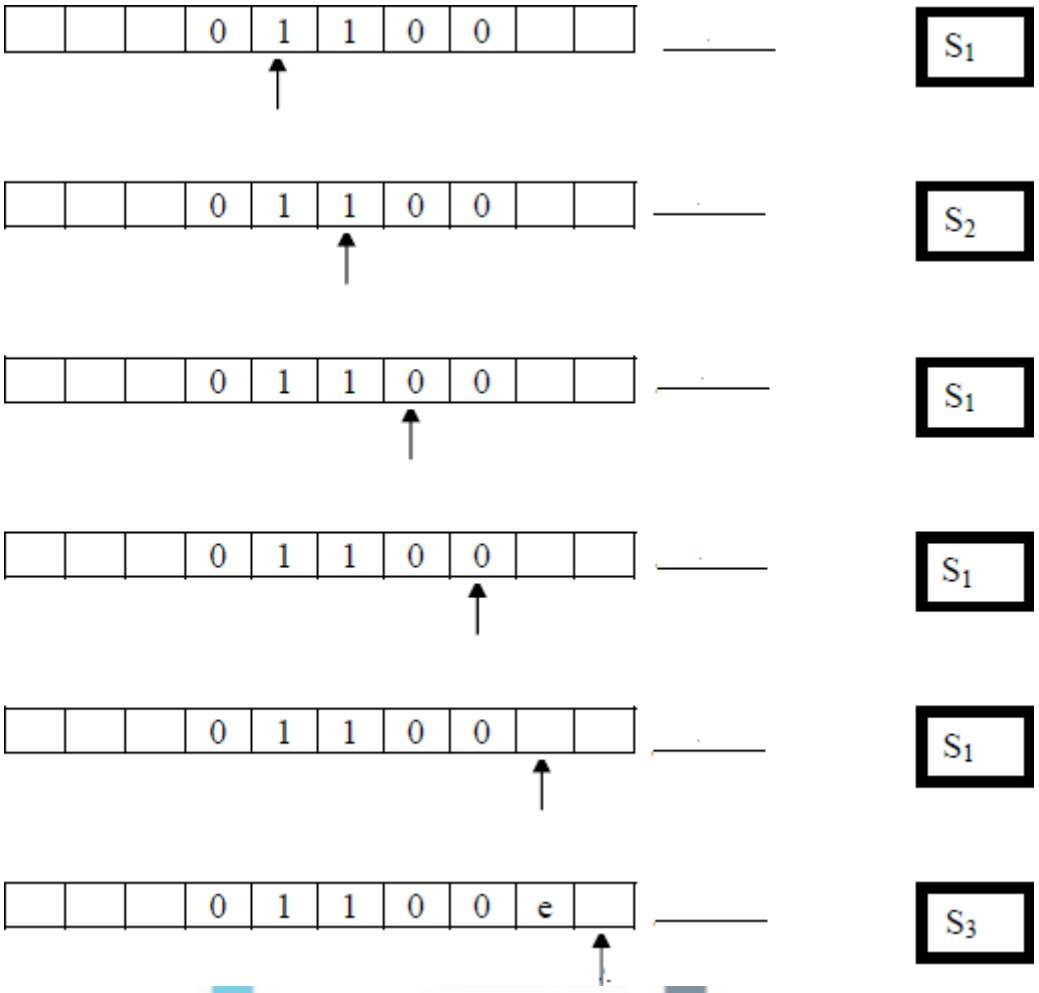
(a)

Number	Correct Label
❶	(0, 0, →)
❷	S_1
❸	(, o, →)
❹	S_3

1 mark for 1 and 3 correct – brackets not required
1 mark for 2 and 4 correct

2

(b)



Mark to end, do not stop at first mistake.
1 mark for first row correct;
1 mark for second row correct;
1 mark for both rows three and four correct;
1 mark for both rows five and six correct;
Must have correct tape contents and current state for each mark

EXAM PAPERS PRACTICE

A answers where the tape has been shifted
DPT for missing read/write head

4

(c) Check if the tape contains an even / odd number of 1s // check parity of number on tape;

1

(d) Turing machines provide a (general/formal) model of computation;
Provides a definition of what is computable // a task is computable if (and only if) it can be computed by a Turing machine;
No computing device can be more powerful than a Turing machine // any algorithm that can be computed by any computer can be computed by a Turing machine;
(The Church–Turing thesis states that) if an algorithm exists then there is an equivalent Turing machine for that algorithm // a Turing machine that can implement the algorithm;
Through the Halting Problem, can be used to prove that some functions cannot be computed;

Max 2

[9]

Q20.

(a) stationary at floor X / stationary at floor Y (Max 2) ;
moving (or equivalent) up ; moving down ;
full ;
out of order ;
emergency stopped;
door open ; door closed ;
lift has been requested for floor X (Max 1) ;
A 'stationary' for 1 (but not in addition to the above)
A 'moving' alone for 1 (but not in addition to the above)
A anything plausible
R anything which reads like an action / input

Max 3

(b) (i) S1 ;

1

(ii) S1 ;

1

(iii) S2 ;

1

[6]

EXAM PAPERS PRACTICE

Examiner reports

Q1.

Most students were able to obtain some marks on this question but very few obtained full marks. Many answers for 2.2 simply stated that state S11 meant that the postcode was valid which was accurate but not enough to get the mark as this did not differentiate S11 from the states S10 and S21.

Many students were unaware of the order or precedence used in regular expressions and therefore dropped marks on question 2.4 due to not including brackets where they were needed. A sequence has higher priority than OR in regular expressions. The regular expression `ab|cd` would match with the strings `ab` and `cd` not the strings `abd` and `acd`; to obtain this latter functionality the regular expression `a(b|c)d` could be used.

Q2.

Nearly all candidates gained full marks for part (a) although part (b) was not as well answered.

Q3.

This question was about data transmission and Mealy machines. For part (a) the vast majority of candidates were able to work out the correct parity bit and give suitable start and stop bits.

For part (b), most candidates were able to explain that asynchronous data transmission meant that that the clocks at the sender and receiver were not synchronised, or that there was no common clock. However, the purpose of the start and stop bit were poorly understood. The start bit is used to temporarily synchronise the clock of the receiver to that of the transmitter. The least well understood part of the question was the purpose of the stop bit. The stop bit serves two purposes. The first is to allow the receiver to process the received data, for example, to transfer it out of a receive buffer, before the next transmission is received. The second is to allow the identification of the next start bit, which is why the stop and start bit always have different binary values.

EXAM PAPERS PRACTICE

Parts (c) and (d) were well answered, with most candidates being able to use the Mealy machine and then recognise the significance of its output.

For part (e), the advantages of serial communication were not well understood, with many candidates failing to achieve any marks and only a few achieving both. When answering this type of question, candidates need to be aware of the context given in the question. Good responses recognised that a small amount of data was being sent, so data transmission speed was not a significant factor in this system, and went on to discuss problems associated with parallel communication that would not occur if serial communication was used, notably data skew and crosstalk. Marks were awarded for points relating to costs if these were justified, but really the fact that fewer wires were required so this would be cheaper was a very weak point given that the question stated that the two communicating devices were next to each other. Candidates need to ensure that they consider their response in relation to the context of a question.

Q4.

This question was about different formal methods of defining a language. For part (a) some candidates correctly identified that a syntax diagram had been used, but many

incorrectly identified the notation as Bakus-Naur Form (BNF).

Three quarters of candidates identified that language definition 3 was different to the other two for part (b) and a pleasing number were able to explain that for definition 3 the sign before the binary number was optional in response to part (c). Some candidates gave an almost correct answer to part (c), recognising that the difference related to the sign, despite having identified the wrong language definition in part (b).

For part (d) candidates had to write a regular expression to recognise the same language as the finite statue automaton on the question paper. This question part was quite difficult and was poorly tackled. The most common mistake was to assume that any repetitions of ba always had to be made before any repetitions of c and to give an answer like a(ba)*c*.

Q7.

Most students were able to answer parts (a) – (d) and (f) well. As was the case for the fixed point binary question on the 2014 paper, a significant number of students did not read part (e) carefully and did not use three bits before and five bits after the binary point. For part (h), most students were not able to describe how a logical bitwise operator could be used to convert ASCII codes into their equivalent numeric character, although some students got one mark for stating a logic gate that could be used. While many good descriptions of how Hamming code could be used to detect and correct single-bit errors were seen, most students provided vague descriptions and a number gave a description of the simple parity bit system instead. Another common mistake was to describe how the bit pattern would be created (rather than checked). Most students answered parts (l) and (m) correctly but found parts (n) and (o) more challenging.

Q8.

This question was about models of computation. Question parts (a) and (b) were both extremely well answered with almost all students demonstrating a basic understanding of Finite State Automata by being able to complete the transition table and identify the state correctly.

For part (c) many students were able to identify that there was no way for any string which caused the FSA to enter state S_6 to ever move the FSA to a different state, but to achieve the mark students needed to explain that this was to prevent invalid strings from being accepted, and only around half of students did this successfully. Students sometimes stated that the state would cause the machine to stop or described it incorrectly as a halting state.

Students' understanding of regular expressions has improved significantly during the lifetime of this specification, and it was pleasing to see in part (d) that approximately three quarters of students wrote a fully correct regular expression. Common mistakes were to use the $^+$ operator instead of the * or to misunderstand the scope of the * operator.

For part (e) students were required to explain why a Turing machine was more powerful than an FSA. Good responses recognised that the key difference was the infinite length tape that the Turing machine has, which could be used as an unbounded memory. Whilst an FSA can use its states as a form of memory, this is by definition finite. A commonly seen but incorrect response was that no model of computation could be more powerful than a Turing machine. This explained why a Turing machine was at least as powerful as an FSA, but not why it was more powerful.

Q9.

Overall, students demonstrated a good understanding of the use of regular expressions.

EXAM PAPERS PRACTICE

(a) For this part, students needed to explain that the expression would accept any string that consisted of zero or more b characters followed by a c. Common mistakes were to confuse the + and * operators and therefore to state that one or more b characters would be required, and to apply the * operator to the c to its right rather than the b to its left. Some students wrote “any number of b characters” which was not enough for a mark. It had to be made clear that zero or more characters were required to achieve the mark.

(b) For this part, students needed to explain that the expression would accept any string that consisted of zero or one b characters followed by a c. Many candidates simply listed the two strings that would be accepted, which were bc and b, which was an acceptable alternative response. Some candidates confused the meaning of the ? operator with either the + or the | operators.

Q10.

The topics covered by this question were generally well-understood. Most students were able to answer parts (a)-(c) well, with the most common error being not reading the question carefully and using a different number of bits from that specified. For part (d), a significant number of students seemed to have memorised that 255 was the largest positive number that can be represented using 8-bit unsigned binary integers and gave this as their answer, even though the question was about two's complement binary.

Answers to part (e) showed that a number of students were not familiar with binary subtraction. However, students who knew how two's complement can be used to subtract one number from another were normally able to describe the process clearly and get good marks. Some answers seen made it clear that students thought that a two's complement binary number means a negative number, not understanding that the two's complement system can be used to represent both negative and positive numbers.

Parts (f)-(i) about finite state machines with output were not answered well with a number of students unable to work out the correct output strings even though there were two completed examples given in the question. Most students were able to get some marks for completing the state transition table in part (i) but few obtained full marks.

EXAM PAPERS PRACTICE

Q11.

For (a), the vast majority of candidates correctly identified the states and rules from the Turing machine's transition function.

For part (b), as in previous years, the trace of the Turing machine's computation was very well completed. This was particularly pleasing as the transition function was longer and the trace more complex than on previous papers. Approximately three quarters of candidates achieved full marks for this question part.

For part (c)(i), candidates needed to recognise that the x and y symbols were used as placeholders for 0 and 1 so that the Turing machine could identify how much of the string had been processed to achieve a mark. Many fell slightly short of this by recognising the x replaced 0 and y replaced 1, but not explaining the purpose of this replacement.

For part (c)(ii) the majority of candidates explained correctly that that Turing machine could be used to copy binary strings on the tape. Some candidates missed out on the mark by just explaining what the Turning machine had done in this specific execution i.e. “writing 01" to the tape, rather than explaining its more general function. A small number of candidates gave answers taken from previous mark schemes which did not relate to this question.

Q12.

Most students did very well on this question with many getting full marks. The most common mistakes were caused by not reading the question carefully and giving either an answer for part (b) that was one of the examples used in the question paper or a permutation that consisted of more than three inputs.

Q13.

Part (a): Approximately three quarters of students correctly recognised that Backus-Naur Form had been used. There was a considerable variation in the spelling of the term but, as long as the meaning was clear, spelling mistakes were not penalised in this question part.

Part (b): The vast majority of students correctly identified which of the two statements was valid.

Part (c): This question part introduced the idea of using a parse tree to demonstrate that an expression was valid. How the parse tree was built was explained in the question and the tree was partially completed to further aid students. Most students made a reasonable attempt at completing the rest of the tree, with nearly half achieving at least two of the three available marks. The most commonly made error was to treat “21” a single digit instead of decomposing it into two digits.

Q14.

Part (a): The relationship between bandwidth and bit rate was well understood. A small number of students failed to achieve the available mark because they simply defined the terms instead of explaining how the bit rate was determined by the bandwidth.

Part (b): There were many good responses given to this question part. Credit was awarded to responses that were either given in the context of data communication or were generic definitions of latency. In context, latency is the time delay between when a signal is transmitted and when it is received. Some students clearly knew that latency related to time, but gave responses that lacked technical accuracy, such as, “the time delay between data being sent and a reply being received,” or, “the time taken to transmit.”

EXAM PAPERS PRACTICE

Part (c): The correct relationship in the example was that the bit rate was double the baud rate. Somewhat disappointingly, just under half of the students recognised this, with a small number stating the relationship the wrong way around.

Part (d): The diagram of the Moore machine was very well completed, with the vast majority of students achieving all four marks. A small number labelled the states but forgot to label the transitions.

Q15.

Part (a): Most students got at least one mark for this question part, usually for identifying that the first required response was “write a program”. Many also went on to achieve the second mark too for identifying that the program being tested would not be run.

Part (b): This question part was poorly attempted. Students needed either to explain that the Halting problem was an example of a non-computable problem, or to state that there was no solution to the problem, so inspection alone could not always determine if a program would halt on a given set of inputs. Some students stated that the Halting problem could be used to show if a program / algorithm was computable or not. This is not the case, as there is no solution to the Halting problem. Rather, it can be used to

demonstrate that some problems are not computable as it is an example of one such problem. A small number of students confused non-computability with intractability.

Q16.

Both question parts were well attempted, with approximately four-fifths of students getting the mark for part (a) and two thirds for part (b). Almost all students knew how a regular expression should be formed. The most common mistake was to confuse the use of the * and + operators.

Q17.

Part (a): The overwhelming majority of candidates were able to correctly complete the transition table.

Part (b): The double circle indicates an accepting state. If the Finite State Automation (FSA) is in an accepting state when it finishes processing the input then the input string is accepted, otherwise it is rejected. Many candidates were confused between an accepting state of an FSA and a halting state of a Turing machine and incorrectly stated that the FSA would halt as soon as it entered the accepting state.

Part (c): The vast majority of candidates got both marks for this question part, indicating that they were able to follow the processing of a string by an FSA, even though they struggled to answer question part (b).

Part d: This question part was well answered with many candidates correctly responding that the FSA would accept strings which began with a 0 and contained an odd number of 1s. Some responses were too vague such as, "accepts 0s and 1s". The most common error was to state that, to be accepted, a string would have to begin with 01 when in fact the second digit could be a 0.

Q18.

For question parts (a)(i) to (a)(iii) answers by example were accepted so long as they included a sufficient number of examples to show the pattern clearly and (when appropriate) there was some indication that the pattern continued beyond the examples listed.

EXAM PAPERS PRACTICE

Part (a)(i): This regular expression would match strings consisting of one or more “a” characters followed by a single “b”. In responses to this type of question, candidates should ensure that their answers make the order of characters explicit. “Strings containing one or more as and a b,” does not make clear that the “a” characters must be before the “b”.

Part (a)(ii): This regular expression would match the strings “ab” and “b”. Some candidates mistook the ? symbol to be a wildcard that would match any single character between an “a” and a “b”.

Part (a)(iii): This regular expression would match a sequence of zero or more occurrences of “ab”. The response, “any number of as and bs,” was not sufficient to be awarded a mark as it did not make clear that the “a” and “b” characters must come in pairs. Candidates who answered by example also have to make clear that the empty string would be accepted.

Part (b)(i): This was the best answered part of question, with most candidates correctly writing a regular expression that would work.

Part (b)(ii): Most candidates achieved one mark for this question part – for the correct start and end of the string. Writing a correct expression for the middle part of the string was more difficult. The correct regular expression was 10(0 | 1)*01. Candidates who wrote an expression that would match the same set of strings, such as 10(0?1?)*01 or 10(0 | 1 | 01 | 10)*01 also gained full credit.

Q19.

Part (a): The vast majority of candidates scored both marks for this question. Candidates who only scored one mark usually named the states correctly.

Part (b): Most candidates gained some of the four available marks and over half gained all four.
Some chose to keep the position of the tape head fixed and move the tape, rather than vice-versa which was perfectly acceptable.

Part (c): The Turing machine outputted 'e' if the tape contained an even number of ones and ‘o’ if the number of ones was odd. Determining this was a difficult task given the limited trace that candidates were asked to complete. Nevertheless a quarter of candidates were able to do this.
Many who did not get the correct answer had managed to understand that the use of the Turing machine related to evenness but that this was whether the number itself was odd or even. A commonly made mistake was to assume that the output of the Turing machine depended only on whether the last digit read was a 0 or 1.

Part (d): It was pleasing to see that many candidates understood that a problem is computable if and only if it can be computed by a Turing machine. Some went on to make a further point, such as that no computer could be more powerful than a Turing machine, but answers scoring both marks were rare.

Q20.

Again, this topic was new to the specification, but apparently well understood by most candidates. The candidate's wording given for question (a) had to make it clear that the candidate was describing a ‘state’; hence ‘up’ would not have scored a mark. However, ‘going up’ gained credit. Some candidates' misunderstanding was manifest when they wrongly described some action by the user of the lift.

EXAM PAPERS PRACTICE