Cambridge International AS Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary Level

COMPUTING **9691/31**

Paper 3 Written Paper **May/June 2016**

MARK SCHEME

Maximum Mark: 90

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2016 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

1 (a) The rule is defined is terms of itself **[1]**
Rule 6 **[1]**

(b) (i) `dog`
This is a `<noun>` using rule 3 **[1]**
A `<noun>` is a `<nounPhrase>` using rule 4 **[1]**

(ii) `a puppy sat`
`<article><noun><verb>` **[1]**
puppy is not a valid `<noun>` **[1]**

(iii) `a cat slept the snake` **Max [4]**

```
<article><noun><verb><article><noun>
  1      3     2     1    3
<nounPhrase><verb><nounPhrase>
      4                 4
<verbPhrase><nounPhrase>
       5
<sentence><nounPhrase>
      6
<sentence>
     6
```

1 mark for each key-change row

(iv) `<adverb> ::= quietly | quickly | slowly` **[1]**
`<verbPhrase> ::=` **[1]**
`<nounPhrase><verb> | <nounPhrase><adverb><verb>` **[1]**

2 (a) The table has a repeated group of attributes //
`BandName` + `Genre` + `NumberInBand` + `SetFee` are repeated for each manager **[1]**

(b) (i) Many bands are managed by one manager // many-to-one **[1]**

(ii) The primary key `ManagerName` in the `MANAGER` table **[1]**
links to foreign key `ManagerName` in the `BAND` table. **[1]**

(c) There are non-key attributes which are dependent on only part of the primary key **[1]**
`Genre + NumberInBand` and `SetFee` will be known from only the `BandName` **[1]**

(d) (i) Issue 1
The Booking table now includes an `AgreedFee` attribute **[1]**

(ii) Issue 2
The booking table now records a `BookingTime` **[1]**

(iii) Issue 3
There is an additional table `VENUE` **[1]**

(iv)

Table	Primary key	Foreign key(s) (if any)
BAND	BandName	**ManagerName**
MANAGER	ManagerName	
BOOKING	**BandName-BookingDate-BookingTime**	**BandName VenueName**
VENUE	**VenueName**	

3 (a) (i) `x a b + /` **[1]**

(ii) `p 2 ^ 2 q + 3 / +`

(1) (1) **[2]**

(b) `3 * (a + b + c + d – e)`

(1) (1) **[2]**

(c) **[8]**

RPNString	ThisChar	StackContents	Temp	INFIXString
x y +	x	x		
	y	y x		
	+	y x		
		x	y	
		x		y
		x		+ y
			x	
				x + y
				(x + y)
		(x+y)		

4 **(a)** Byte 1: `31` (1)
Byte 2: `-63` (1) **[2]**

(b) 93 07 **[2]**
1 mark per byte

(c) 6A F5 **[2]**
1 mark per byte

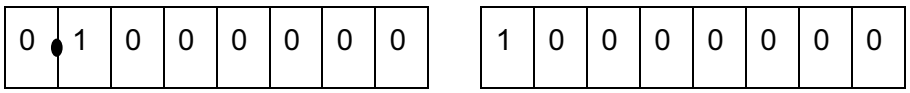
(d) **(i)** The mantissa starts with a 1 digit **[1]**

(ii) Mantissa: −1 + 5/16 // − 11/16 // − 0.6875
Exponent: 11 **[2]**

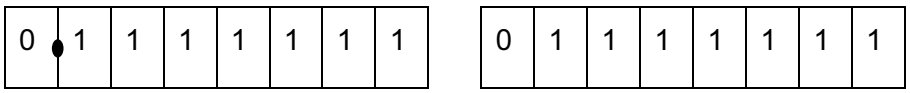
(iii) −11/16 * 2^11 // −11 * 2^7 // −1408 **[1]**

(e) The mantissa starts with 10 // the first two bits of the mantissa are different **[1]**

(f) Smallest **[1]**



Largest **[1]**



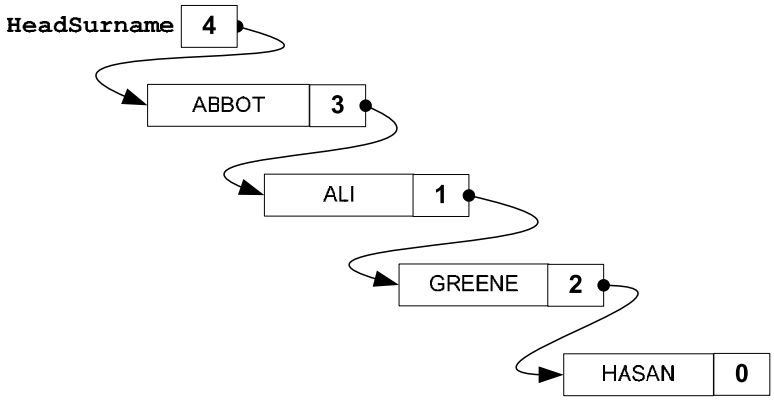
5 **(a)** **(i)** Dynamic data structure changes size at execution time **[1]**

A static data structure has a fixed size **[1]**

(ii) Dynamic data structure matches the size to data requirements // **[1]**
Takes memory from heap as required //
returns memory as required (following node deletion) //
There is no wasted memory space / makes efficient use of memory

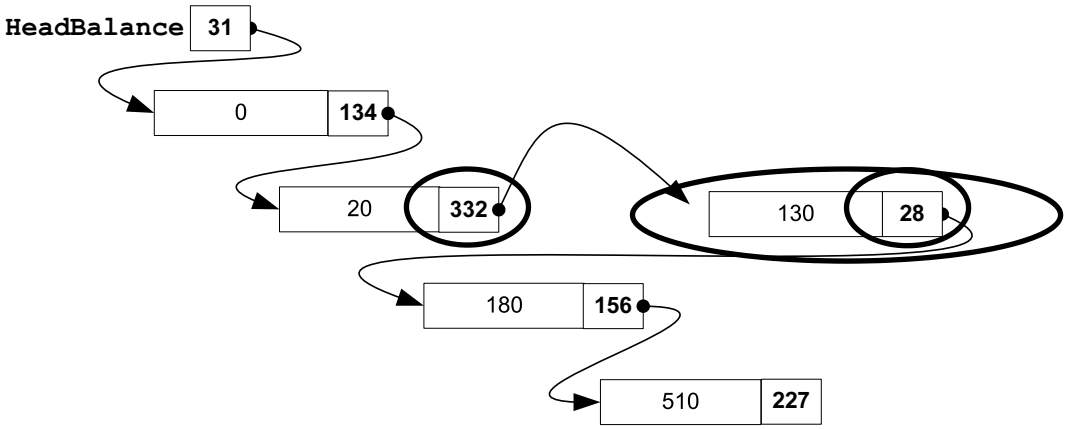
(b) ABBOT (1)
1 (1) **[2]**

(c) **[3]**



Mark as follows:
HeadSurname = 4 (1)
Names in correct order (1)
Link pointers correct (1)

(d) (i) **[3]**



Mark as follows:
New node inserted (1)
332 for the correct node (1)
28 for the correct node (1)

(ii) Start at the head pointer (1) **Max [4]**
Follow the link pointers until (1)
The value found is greater than the value to insert (1)
Pointer of previous item points to new item (1)
New item pointer points to next item in the list (1)

6 (a) *A production line paint sprayer*

"The robot is a mechanical device":
The chassis and robotic arm are a mechanical device **[1]**

"Movable"
The paint sprayer arm must be able to position correctly to spray all parts of the car **[1]**

"Can sense its surroundings"
Sensors will sense when a car is in position // determine when an obstacle is encountered / edge of the car is reached **[1]**

"It is a controlled by a computer program"
e.g. The computer program sets the parameters/type of car/paint to be used **[1]**

(b) Robotic arm **Max [4]**
To position the spray nozzle to the correct position

Sensor
Capture data

Actuator // **Motor**
To drive various motors to perform the robot's movement

Microprocessor
To process the various inputs and execute the control program

Memory
To temporarily store input data // store program

Speaker // **bleeper**
To provide audio output

Any 2 × 2

7 (a) (i) The program as written by the programmer // the program written with the text editor **[1]**

(ii) The output from the compiler // the program in machine code / byte code / intermediate code **[1]**

(b) (i) All the keywords which make up the syntax of the language **[1]**
A token for each keyword **[1]**

(ii) `DECLARE, CONSTANT, CALL, WHILE` (any **three**…) **[1]**

(iii) A list of all the identifiers used by the programmer. **[1]**
A pointer to where their value is stored in memory **[1]**

(iv) `Counter, Jobs, Position, ChangeRate, InitialiseGrid` (any **three** …) **[1]**

(c) *Lexical analysis …* **Max [5]**

Remove any whitespace from the source code (1)
Remove any comment statements (1)
Check for obvious errors in the use of identifiers, e.g. they do not exceed 64 Characters (1)
Replace all language keywords with a token (1)
Add all identifiers to the symbol table (1)
All identifier names are replaced in the code by a pointer value (1)

(d) (i) Altering the object code so that it runs faster // takes up less memory **[1]**

(ii) The lines that have the expression `x + y` **[1]**