2.1 Data structures, abstract data types part 2 Mark Scheme

Q1.

(a) (i) First In First Out;
or by description

(ii) Last In First Out;
or by description

2

(b)

	FIFO	LIFO
Queue	✓	
Stack		✓

2

(c) Reverse the contents of a queue/list;
Push all contents of queue/list onto stack then pop them off into a new queue/list;
Procedure/function calls;
Local variables;
Parameters;
Return Address;
Volatile environment; **A** register contents State 1 Describe 1

2

(d) List of elements inserted into tree;
To allow rapid/fast searching of the data;
To output sorted/ordered data;

2

[8]

Q2.

(a) *1 mark for each correct entry*

New	*Last*	*Ptr*	*Values*				
			[1]	*[2]* ***max***	*[3]* ***6***	*[4]*	*[5]*
6	*3*	*1*	*4*	*7* ***max***	*9* ***1***		
		2					
	2					*9*	
	1				*7*		

				6			

6

(b) Insert 6/a value into the array/ in the correct position;

1

[7]

Q3.

(i) Allow addresses in the Pointer column.

Position	Name	Running Time	Address	Pointer
1	Process6	7	01400	4 (02300)
3	Process7	17	01700	5 (04100)
4	Process2	17	02300	3 (01700);
5	Process9	45	04100	-1; A 0;
6	Process5	2	01200	8 (01900)
8	Process19	5	01900	1 (01400);

1 mark for 4,5,3 correct

1 mark for null pointer correct A sensible const. Name representing null pointer

1 mark for 8,1 correct

3

(ii) Array; of records; *OR* linked list; of records; *OR* 4 1-D arrays;
One for each column; *OR* one 1-D array for process name;
One 2-D arrays for numerical data;

2

(iii) Marks to be allocated as follows:

1 for initialisation	ListPointer ← HeadPointer;
1 for while not at end of list	While ListPointer <>-1 Do;
1 for printing	Print ListArray[ListPointer].Name;
1 for getting next pointer	ListPointer ← ListArray[ListPointer].Pointer;
P1 if headpointer is reassigned	

Any name acceptable for ListPointer and ListArray
Note: a sorting method gets a maximum of 3 marks (inefficient)

Alternative solution
REPEAT UNTIL next=-1 OR IF listpointer <>-1 then REPEAT..

4

(iv)

List	*Reason*
List of suspended/blocked/halted/ unrunnable processes;	waiting for a resource or complete a requested I/O transfer;
List of inactive/dormant jobs;	Waiting to be admitted to the system;

I currently running processes
I interrupt

2

[11]

Q4.

(a) *Any two at two each; If entrance method doesn't match exit method mark one wrong and the other correct*
R Voice **R** Written to ticket
Computer system/Printer prints number on ticket at entrance;
Driver types number into system using a keypad at exit barrier;

Computer system encodes number on a magnetic stripe on ticket at entrance;
R Magnetic card
Ticket number read by a magnetic stripe reader at exit//inserted into a magnetic stripe reader at exit; **A** magnetic strip/stripe scanner

Computer system/Printer prints number printed on ticket at entrance;
Number read by an optical character reader/OCR at exit//ticket inserted into an optical character reader at exit;

Computer system/Printer prints number in barcode form on ticket at entrance;
Number read by barcode reader at exit//ticket inserted into barcode reader at exit;

Computer system/Printer at entrance punches holes on ticket which are a coded form of number//Kimbal tag produced at entrance which encodes number;
Number read by sensor (mechanical or optical) at exit//ticket inserted into sensor at exit//Number read by Kimball tag reader at exit;

Computer system/printer prints number using magnetic ink;
At exit MICR reader reads number;

Computer system/printer prints marks (encoding number) on ticket;
At exit, OMR device is used;

4

(b) **R** any other data types. Mark is for field name + correct data type.

NB synonyms for RandomNumber must include Number, e.g. IDNo, TicketNo, Number. **A** RandomInteger, **R** e.g. Vehicle ID **A** VehicleIDNo

A DateTicketWasIssued

```
Record

  RandomNo : Integer;
```

R anything else

1

```
CurrentDate :
    String/Date/DateTime/TDateTime/TDate;
```

1

```
 ArrivalTime :
String/Integer/Time/DateTime/TDateTime/TTime;
```

1

```
 LengthOfTime/LengthOfStay/TimeStayed : Integer;
```

R anything else

```
  Cost/AmountToPay : Integer/BCD;
End;
```

1

1

A Alphanumeric for String
R Text **R** LeavingTime **R** Binary,Byte,LongInteger
R Date for FieldName
R Date/Time but don't penalise twice

[9]

Q5.

Queue is FIFO ; (1)
Stack is LIFO; (1)
Given that:
Process of taking elements from queue to stack(1)
Process of popping stack(1)

[4]

Q6.

(a) Head (Tail (Days)) = Mon
R [Mon], MON (1)

Tail([Head(Days)]) = [ ] (1)

Empty(Tail(Tail(Tail(Days))))=False (1)

3

(b) Elements in a list can only be accessed sequentially;
elements in an array can be accessed directly;
using the subscript;

Any 2 points

2

[5]

Q7.

(a)

Ptr	Temp	List									
		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
		43	25	37	81	18	70	64	96	52	4
1	43	25	43								
2	43		37	43							
3											
4	81				18	81					
5	81					70	81				

6	81						64	81			
7											
8	96								52	96	
9	96									4	96
10											

Ignore Ptr & Temp column
1 mark for each of rows 1, 2, 4, 5, 6, 8, 9
(Final list 25, 37, 43, 18, 70, 64, 81, 52, 4, 96)

7

(b) Control will pass to the instruction after Endwhile;
/the instruction/command/statement after Endwhile will be executed;
Program will exit while-block; loop stops;
A algorithim stops; **R** program stops;

Max 1

(c) (i) 25;

If part (a) not fully correct allow follow through: or lower of [1] & [2]

3

(ii) 81;

Only allow follow through mark if the list at the end of part(a) is still a partially sorted list

(iii) 96;

Must be 96 in all cases

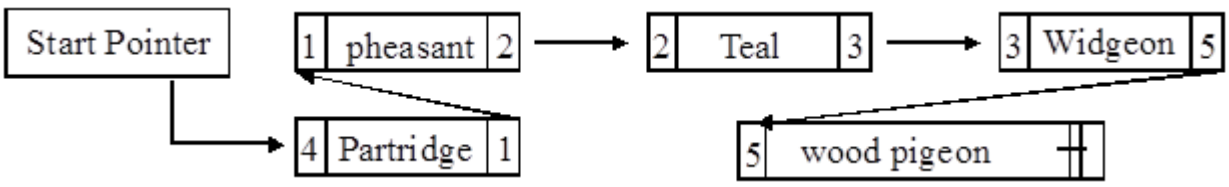
[11]

Q8.

(a) (i) Data must be in given order

	1	pheasant	2
	2	teal	3
	3	widgeon	5
START	4	partridge	1
	5	woodpigeon	0

//



End Pointer can be blank
1 for correct START and END pointers;

1 for correctly numbered nodes and correct pointers (Need all birds);

(ii)

	1	pheasant	7
	2	teal	3
	3	widgeon	5
	4	partridge	1
	5	woodpigeon	0
START	6	grouse	4
	7	snipe	2

// correctly amended diagram

1 for grouse and snipe physically at end;
1 for correct pointers (if not as ms than clear and logical);

2

(b) The amount of memory taken up can vary;

// The size / length of the structure / linked list can vary;

1 mark

at run time;

1 mark

2

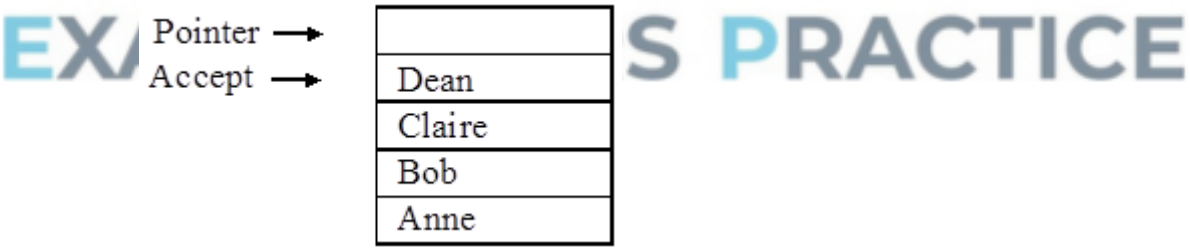
(c) A heap / stack/ a pool of available locations;
A pointer holds the address of the allocated block / next available location;

2

[8]

Q9.

(a)



2

Dean accessed first
Pointer / arrow

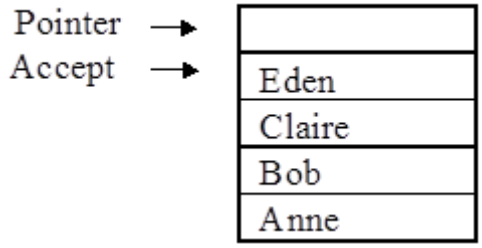
	Dean	Claire	Bob	Anne

End pointer ↑ Start pointer ↑

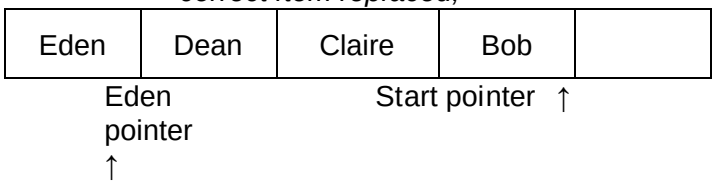
Anne accessed first;
Named pointers correct;

2

(b)



correct item replaced;



Correct item retrieval & input;
Correct moving of pointers

3

(c) In a linear queue data is static, so queue 'moves' through storage/
In a FIFO structure storage locations are only used once;(1)
In a circular queue, the locations will be re-used;(1)
Thus a circular queue has a more efficient use of memory;(1)

1 mark for each of 2 points

2

[9]

Q10.

(a) Magnetic Strip(e) Reader (Not swipe card reader);

1

(b) What: Extra/last digit(**A** number) in borrower code;
Calculated digit;
Why: To detect if data/code has been corrupted;
To check that data/code is valid;
To ensure integrity of data/code;
A To check that number is valid
A To check that data/code is still correct after transmission
R To check that data is plausible

2

(c) Reason: magnetic stripe reader may not be able to read borrower code;
Because magnetic stripe is damaged;
Can phone in code;
Code needs to be entered through a keyboard;

1

(d) Bar code reader/Bar code scanner;
R Scanner
R Light pen

1

(e) (i) Unique field of a record/field used to identify record;

1

(ii) BookCode;

1

(iii) Serial;

1

(f) (i) BookCode;

1

(ii) Reason: Any one for 1 mark
So one pass is possible;
Reduce time taken to update Books file;
Saves time;
Order: Same as Books file/ordered on BookCode;

2

(g) **NB** steps must be clear. **R** a narrative in which steps not made explicit = zero marks

Alternatives:
Compare Current Date with DateBookToBeReturnedBy field;
If = or >= or >; If (LoansStatus = OnLoan);
And TodaysDate >|>=|= ;
; DateBookToBeReturnedBy

Steps:Open Books file;
(Read or idea of going to) each record in turn/
(Read or idea of going to) next record;
Until EOF;
If LoanStatus = OnLoan;
Then
Compare DateBookToBeReturnedBy field with current date;
If = (allow <= and <) ;
Then Write details to OverDueBooksFile;

4 marks

Data:Any three × one each (No T.O.)

BookCode;
BorrowerCode;
DateBookToBeReturnedBy;
ISBN;
Title:
Author;
***R**. any others Candidate must state these accurately*

3 marks

7

[18]

Q11.

(a) Tail(Ports) - [Barcelona, Athens, Alexandria, Tunis, Lisbon]

square brackets needed

1

Head(Tail(Tail(Ports))) – Athens (2)

[Athens] (1)

2

Empty(T(T(T(T(T(T(Ports)))))))) – True (2)

True [ ] (1)

[True] (0)

2

(b) **Recursively defined**
A definition which is defined in terms of itself/contains within its body a reference to itself/calls itself ;
A re-entrant; (In specimen papers 2001/2, but refers specifically to a procedure)

1

(c) **Stack necessary**
The state of the machine/contents of appropriate registers/ return address // saved each time the procedure is called (1) and retrieved in reverse order from the stack as control is progressively returned (1)
OR
Different value of parameters /local variables (1) must be available each time procedure is called (1)
OR
P must be re-entrant (In specimen papers 2001/2)(2)

2

(d) Lisbon first (1)
Southampton last(1)
All 6 in order (1)
No punctuation (1)

i.e. Lisbon Tunis Alexandria Athens Barcelona Southampton;

4

(e)

1	Southampton	5	
2	Barcelona	6	Head Pointer
3	Athens	2	4
4	Alexandria	3	
5	Tunis	0 - terminator	
6	Lisbon	1	

2

[14]

Q12.

(a) (i) VAR/CONST/TYPE/DIM/FUNCTION/PROCEDURE/LABEL
Or similar, name and type;; keyword and name;;

2

(ii) Eg x:=5 / y ← y – 1

1

(iii) Example of IF / CASE / SWITCH statement

(1 mark for keyword, 1 mark for selection criteria)

2

(b) (i)

	S		S	
	D	[5]	M	[5]
	K	[4]	T	[4]
	C	[3]	C	[3]
	T	[2]	K	[2]
	M	[1]	D	[1]

1 mark for each correct entry

10

(ii) Algorithm: reverse content of array
R re-arrange

1

[16]

Q13.

(a) Optical Mark Recognition/Reading. (Not Optical mark reader)

1

(b) Extra digit added to the transaction code (1)
To detect if data has been corrupted (1)

2

(c) (i) Unique field of a record/filed used to identify record

1

(ii) Transaction code

1

(iii) Not indexed sequential / Serial or Sequential (1)
Because all the records have to be examined (1)

Or

Direct access based on a hash code of the chosen numbers(1)
Only a few records will need to be checked (when collisions occurred)(1)

Max 2

(iv) Random or direct access(1)
Record can be located by simple transformation of transaction code /hashing technique used/algorithm used to store and retrieve records(1)
Indexed sequential with transaction code as key field(1)
Rapid access via the index is possible to find the necessary record(1)

Max 2

(d) *Any 4 points × 1 each*
Ticket scanned/ Read ticket
Check digit used to check accuracy of scanning Ticket validated, (e.g. not out-of-date, draw not yet made)
Operator informed if ticket does not scan/is invalid
Transaction code sent to central computer
Correct file selected
Ticket's record found/Look up ticket's record/Look up record with given transaction code
Get draw date from transaction record
Get numbers from system (for the correct draw date)
Ticket numbers checked against draw
If a winning ticket prize money determined
Result sent to point of sale machine
Result displayed at point of sale machine

Max 4

[13]

Q14.

(a) Array must be sorted (1), on the field being used as the search key (1)

2

(b) Description must include the following points: Find median record of array **(I)** Compare key field of record at median position with required search key, exit if found (1) If search key lower (i.e. required record in first half), discard second half, else discard first half (1) Repeat process (1) until either found, or no further division possible so record does not exist (1)

5

(c) On each iteration, half the possible matches are eliminated, compared with only one for the linear search (2)
Linear search on average scans n/2 records, compared with $\log_2 n$ which is smaller "Looks at fewer records" without further explanation (1)

2

[9]

Q15.

(a) Causes process to repeat indefinitely

NOT repeats until maintain is TRUE

1

(b) Maintain has two values, TRUE / FALSE (1), => must be Boolean (1) n is used as an array subscript (1) => must be integer (1) or n is used as a loop control and can never be non-integer within the algorithm (1) => integer (1)

NOT numeric - too vague

4

(c) See table for model solution 1 mark each indicated section completed correctly, including follow–through (7x1); additional 1 mark for correctly modifying n downwards in penultimate section If candidates go completely wrong but clearly deserve some credit marks can be awarded on the following criteria, up to a maximum of 2 marks for correct sequence of loop repetitions, including the change from 6 to 5 then 6 - i.e. the column for n, including correct exit 2 marks for correct completion of the sequence of stations, ie the org, dest, start, finish columns

2 marks for correct completion of totalkm column, i.e. correct lookups and totalling 2 marks for correctly executing inner if branches, i.e. setting maintain and resetting totalkm in correct places. Total 8 marks for all-correct trace follow-through marks should be awarded where appropriate

8

(d) 2 marks for diagram, or explanation, showing that the journeys indicated above cover all routes in both directions *marks can be awarded for any reasoned answer (indicating achievement or not) providing it is consistent with the candidate's trace table* Note: the sequence is MK -> SW -> CW -> SW -> TW -> HK -> MK -> QB -> SW etc., which does cover all lines in both directions. Strictly speaking, whether the objective is achieved depends whether journeys to/from MK depot are passenger-carrying / revenue-earning or not. Either interpretation is acceptable - the marks are awarded for the explanation.

n	org	dest	last	start	finish	totalkm	maintain	Remarks
	0	3	1					
							FALSE	
				MK				
					SW			
						15		
	3							
0								

Given

1								
		4						if ignored
				SW				if ignored
					CW			
						+27 =		

						42		
	4							N<6 so rpt

1 mark

2								
		3						if ignored
				CW				if ignored
					SW			
						+27 = 69		
	3							N<6 so rpt

1 mark

3								
		1						if ignored
				SW				if ignored
					TW			
						+37=106		
	1							N<6 so rpt

1 mark

4								
		5						if ignored
				TW				if ignored
					HK			
						+34=140		
	5							N<6 so rpt
5								

1 mark

		2						if ignored
		0						>140 so if executed
			5					
							True	
				HK				
					MK			
						+12=152		
	0							N<6 so rpt

1 mark

6								TRUE so if executed
5								
						0		
							False	
		2						
								if ignored
				MK				
					QB			
						+28 = 28		
	2							N<6 so rpt

1 mark

6								
		3						if ignored
				QB				if ignored
					SW			
						+43 = 71		

	3							
								N = 6 so stop repeat loop
								end while

1 mark

2

[15]

Q16.

(a) Elephant 4
Deer 1
Bear 5
Rabbit 0
Cow 2

1 mark for rabbit having a pointer of 0
1 mark for the others correct

2

(b) Start = 3
Freestorage = 6

2

(c) Check for free space
Put data into the array at the position indicated by freestorage (animals[6])
Find position where "Monkey" must go in list (between Elephant and Rabbit)
Method for finding position
Alter "elephant" pointer to point to "Monkey"
Make "Monkey" pointer point to "rabbit"
Alter the freestorage pointer to point to next space / to indicate no more free space (0 / –1)

This may be answered as a pseudocode algorithm but any method that makes the steps clear is acceptable
Any 5 × 1

5

[9]

Q17.

(a) Structure shown correctly with data in consecutive locations, and start and stop pointers

Must have data in boxes or a label

3

(b) Procedure insert

```
{Check for overflow}
If (Start pointer=1 and) stop pointer = max size
```

2 marks

```
(or Start pointer = stop pointer+ 1) then
     Report queue full
     End
Endif
```

1 mark

Parts in brackets refer to circular queue not essential. 2 marks for checking. 1 mark for an attempt to check. 1 mark for stopping if full

```
        {Insertion of data}
        (If stop pointer = maxsize then
               Stop pointer:= 1
        Else)
               Stop pointer:=stop pointer+ 1
        (Endif)
        queue(stop pointer):=data
endproc
```

1 for changing stop pointer. 1 for inserting data
2 marks

5

[8]

Q18.

Repeat
S → Q
Until Q empty
Queue emptied to a stack
Elements taken from front of queue and placed / pushed on stack

2 marks

Repeat
Q → S
Until S empty
Stack emptied to a queue
Elements popped/taken from top of stack placed in queue

2 marks

Or suitable diagram

1 mark

[4]

Examiner reports

Q1.

(a) Many candidates scored very well on this question. The terms FIFO and LIFO were very well understood.

(b) Again, a high scoring part of the question. Most candidates managed to place the ticks in the correct boxes.

(c) Quite a few candidates were able to state a suitable example of the use of a stack but few were able to describe it satisfactorily. It should be stressed that candidates should give computing examples. A significant number of answers referred to stacks of plates, books etc.

(d) Once again, many candidates obtained a mark for stating a suitable example but few were able to describe it satisfactorily.

Q2.

(a) This question was very badly answered. Many candidates failed to attempt the question at all and, of those that did, there were very few correct answers.

(b) As so few candidates were able to answer part (a) it was not surprising that very few acceptable responses were received for part (b). What was surprising was the number of candidates who were able work through the algorithm to answer part (a) but were unable to describe its purpose.

Q3.

Many candidates were able to complete the pointer column in the table correctly but could not adequately describe a suitable data structure for this table. Most stated array or linked list but very few noticed that the columns required different data types and therefore an **array of records** or a **linked list of records** or several arrays were required for full marks. In part (iii) many candidates could write a suitable algorithm, some even provided very elegant, recursive, solutions. However, a few candidates reassigned values to the head pointer as they worked their way through the list. This is not appropriate. Others printed the pointer rather than the name. A possible solution gaining full marks would be:

```
ListPointer ‟ HeadPointer
While ListPointer < > -1 Do
  Print ListArray[ListPointer].Name
  ListPointer ‟ ListArray[ListPointer].Pointer
EndWhile
```

In part (iv) very few candidates seemed to remember that the list they were working with in the question was that of runnable processes, and that only one process at any one time can be running, so a list of running processes would not be sensible. This leaves suspended processes (waiting for a resource) and inactive jobs (waiting to be admitted to the system).

Q4.

Many candidates were able to suggest a suitable method at the exit barrier for submitting the number assigned to the ticket to the computer system. Fewer were careful enough to describe how the ticket was assigned to the ticket at the entrance barrier. The examiners were expecting a printer to be referenced or the action of printing in the case of a barcode,

OMR, OCR, MICR and plain text solution for writing a number to the ticket; the action of encoding or writing to the ticket in the case of a magnetic stripe or smart card and the action of punching in the case of Kimball or Kimball-type tags. The better candidates offered such descriptions.

A lack of experience of using a third generation programming language was exposed by part (b). Many candidates gave data types that were lifted straight from Microsoft ACCESS and therefore gained no credit. The question explicitly requested data types that would be available in a third generation programming language. Candidates must understand that their study of this subject must extend beyond products that have been intentionally designed to enable people with no background in Computing, and no desire to be educated in this subject, to achieve practical results in the minimum of time. For very simple practical tasks, this is desirable but as a foundation for more extensive tasks this is a recipe for disaster. Assembling flat-pack furniture is not considered adequate enough to qualify as a cabinetmaker.

Some candidates had difficulty selecting relevant fields and in some cases when they did choose relevant ones lost a mark for poor choice of identifier. For example, several candidates chose *Date* instead of *CurrentDate*. The clue was in the question stem which stated that "the computer system remembers the *Current Date, Arrival Time* and *Randomly generated Number*." A principle of software engineering is that identifiers should be meaningful and reflect the real world entities which they represent.

Q5.

Many candidates failed to score as highly as they should have through their lack of ability 'to organise relevant information clearly and coherently'. The basic argument is that the fact that a queue is a 'First In, First Out 'structure, while a stack is a 'Last In, First Out' structure. This means that if the elements of a queue are pushed onto a stack, their order, when pushed off the stack, is reversed. Incidentally, the word is 'queue', not 'que' or 'cue' – mistakes prevalent in candidates' answers.

Q6.

Parts (a) (i) and (a) (iii) were answered correctly by most candidates as Mon and False respectively, (not [Mon]). However, in (a) (ii) [Head(Days)} is the list [Sun] and the tail of this list is the empty set [ ]. Had these data been stored in a one dimensional array, instead of in a list, each element could have been accessed directly using the subscript, rather than having to be found through a sequential search or using the Head / Tail functions defined.

Q7.

(a) There was a definite improvement in the dry running of a piece of pseudo-code. Many candidates were able to correctly complete the table to gain the 7 marks. Some candidates instantly recognised the bubble sort and decided to skip the dry run and write the final values on the bottom line (the mark scheme penalised this severely). Others assumed that it was a complete bubble sort and simply wrote down the list of numbers in ascending order. However, a significant number of candidates still did not seem to be prepared for dry-running an algorithm. The grid was left blank or was completely filled in with totally irrelevant numbers by such candidates.

(b) This was poorly answered with many candidates stating that the program stopped or that 'nothing will happen'. A few candidates gained credit by stating that control will pass to the instruction after EndWhile.

(c) Those candidates who answered (a) correctly obtained at least 2 marks for this part. Even good candidates often did not spot that another run through the algorithm would put 81 into the ninth element of List.

Q8.

This question referred to the loading of data into a linked list. No marks were given if the items had not been loaded in the given order. Binary trees were not credited.

In part (a), nodes should have been numbered with correct links. A start pointer was expected with some indication of the last item. In (ii) 'snipe' and 'grouse' had to be added at the end of the list and the pointers amended correctly to incorporate them.

In (b), many candidates correctly said that in a dynamic structure the amount of memory required would vary but few gained the second mark here by stating that this would be at run time. Size was an acceptable alternative to amount of memory required, but it was insufficient to say that 'items could be added or deleted'. Incorrect answers included the ability to sort items.

In (c), answers that memory is allocated from a heap or pool of available locations and pointers hold the address of the next free location or of the last allocated block gained both marks.

Q9.

This focused on stacks and queues. Again, marks were lost through lack of care in the diagrams. Two vertical columns of names, with Dean at the top of one and Anne at the top of the other were insufficient to show how this data would be stored in these two different data structures. Full marks were gained by two groups of locations, either vertical or horizontal, with pointers to show the top of the stack and the front and rear of the queue. Most candidates did successfully remove and input the correct items in part (b). In part (c), many seemed to think that pointers were only used in circular queues, and that queues were usually implemented in this way so that 'once an item is retrieved and processed it can be placed back into the queue by adding it at the rear'. Good answers noted that a queue is a 'Last In First Out' structure. Data is removed from the front and added to the rear, so current data in a linear queue will 'move' through memory. In a circular queue, the locations in the front of the queue, which are no longer current, will be re-used. A circular queue thus makes more efficient use of memory.

Q10.

A surprising number of candidates had difficulty recognising the two techniques for encoding data referenced in this question and therefore failed to identify the correct reading device. Both are in common use though candidates may have greater experience of the use of barcodes. Barcodes must be a daily feature of most candidates' lives.

Many candidates identified the last digit in the borrower code as a check digit and many candidates explained that it was a calculated digit calculated from the other digits in the borrower code. However, some candidates could not explain precisely enough why it is used. The check digit is there so that corruption of the borrower code can be detected when it is entered into the library's computer system. Several candidates thought that the check digit checked the whole card, i.e. checked that the card belonged to a valid user of the library.

Part (c) was well answered with the most popular answer being to key in the code if the reader fails to read the code.

The term primary key gave few candidates any difficulty. The better candidates answered part (e)(iii) correctly. The Loans file uses serial file organisation.

Part (f) caused some difficulty for the weaker candidates, many answering from the perspective of the layperson. For example, the commonest incorrect answer for sort field for the Books file was DateBookToBeReturnedBy. In part (g), other candidates ignored the Books file altogether and referred only to the Loans file when stating the processing steps. In fact, in many instances the required processing steps were described not stated, the candidates' responses appearing as a narrative that could not have been used to program a computer. Narratives scored zero marks. Previous reports have made reference to this and the advice given has been similar. If a question asks a candidate for the processing steps then a sequence of steps is required. Candidates are advised to use a linear layout that lists the steps in logical sequence. For example,

```
Open Books file
Repeat
              Read next record
              If LoanStatus = OnLoan
                   Then
                   If DateBookToBeReturnedBy < CurrentDate
                         Then Write Details to OverdueBooks File
Until End Of Books File
```

The majority of the candidature was able to correctly identify the data to be extracted – BookCode, BorrowerCode, DateBookToBeReturnedBy. Some candidates lost marks here because they failed to state these fields accurately. Others failed to gain marks because they invented irrelevant fields. ISBN, Title and Author were accepted because these might well have been present in the Books file.

Q11.

(a) This part asked what result would have been returned by the specified function calls. So Tail (Ports) would have returned [Barcelona, Athens, Alexandra, Tunis, Lisbon] with the brackets being part of the correct answer. Head(Tail(Tail(Ports))) would have returned Athens, without brackets and the answer to the last part was True, not [True] or True [ ].

(b) Most candidates could say what recursively defined was, although some answers were barely sufficient and 'It calls itself' was deemed insufficient.

(c) Explanations of why a stack was necessary to execute procedure P recursively frequently missed the point.

(d) Full marks were gained for this part by an answer of:
Lisbon Tunis Alexandria Barcelona Southampton,
or even of:
LisbonTunisAlexandriaBarcelonaSouthampton,
as no punctuation was printed. A list of:

- Lisbon
- Tunis
- Alexandria
- Barcelona
- Southampton

was also accepted.

(e) Candidates who did not score full marks for 10(e) mainly numbered the ports in alphabetical order, rather than using the pointers to point to the next port in the list, or gave inadmissible or absent end-of-list markers.

Q12.

Part (a) was always attempted, but many candidates confused a declaration with an assignment. Some candidates tried to answer in general terms rather than with examples asked for. In part (b) some candidates got full marks and demonstrated clearly their ability to follow through an algorithm. Many candidates did not seem to understand how to dry run algorithms; or this was not taught in some centres. Many candidates guessed wrongly that it was a sort of algorithm and simply wrote down the letters in alphabetical order into the array. Some candidates thought that the algorithm reversed the letters and then back again. Most candidates who correctly followed through the algorithm then correctly recognised that the order of the letters in the array was reversed.

Q13.

Many candidates correctly identified the method as Optical Mark Recognition. Candidates who answered “Optical Mark Reader” referred to the device, not the method, and so were not rewarded. Some candidates answered incorrectly that a check digit checked that the chosen numbers were unique. An answer that stated that a check digit is an extra digit added to the transaction code obtained a mark. The mark scheme allocated a second mark to an answer that stated that the check digit was used to detect if data was corrupted. The emphasis was on error detection, hence the non-specificity in stating what was being corrupted. Several candidates went into detail and described how the check digit is calculated using a modulo- II method. This was really answering more than was required for one mark, as this response described both what it is and how it is generated.

The majority of candidates correctly defined the term ‘primary key’. However, several of these candidates then incorrectly identified “Point of Sale Identification Code” as the primary key for the transaction records instead of “Transaction Code”. These candidates appeared to lack an understanding of the term ‘transaction’.

In part (c) (iii) the better candidates realised that all the records have to be examined to find the ticket(s) with the winning numbers. These candidates then showed good knowledge of file organisations by answering “serial”. Weaker candidates seemed to be unfamiliar with the term ‘file Organisation’, responding with answers such as “use a database or spreadsheet”. Other successful candidates answered that if each transaction’s chosen numbers were hashed, then the generated address could be used to store each transaction’s details. After the draw, the winning numbers could be hashed to locate the transaction(s) that had won.

In part (d) many candidates did not appreciate the detail of the processing steps that have to take place if the computing system is to check if the ticket is a winning ticket. These candidates showed a distinct lack of insight into the operation of the computer. Their answers were superficial and from the perspective of the ticket holder not the computer system. The better answers were on a different analytical plain. These answers used appropriate technical terms e.g. ticked “scanned”, “check digit” used to check accuracy of “scanning”, “transaction code” sent to “central computer”, correct “file” located/ “record” with given “transaction code” found or “transaction code” compared with winning “transaction codes”, etc. Compare this with an answer pitched at the level of a ticket holder. “The numbers on the ticket are entered. The computer system checks if these are the winning numbers. The user is informed”. Such an answer gained no marks.

Q14.

Knowledge of sort procedures seems good, but the ability to express it with anything like the precision of language expected at A-level is not. In particular, far too many candidates treat file, record and field as interchangeable terms, which is unacceptable at this level.

For part (a), nearly all candidates pointed out that an array needed to be sorted for a binary chop search to work, but very few realised that, say, an array with ten fields could be sorted in ten different orders and only one would work - that in which the sort key was the field being searched.

For part (b), most scored reasonably well, although as usual descriptions were full of waffle. Candidates seemed to be trying to paraphrase an algorithm they had been taught - while formal algorithms are not in the syllabus for this paper, if a candidate wants to present an answer in pseudocode or even flowchart form it will receive full credit. A common mistake was to say that the search key was compared to “the middle record” (rarely the more accurate “key field of the median record”) and one half or the other discarded, ignoring the possibility that it might be matched enabling the search to end. Many candidates failed to indicate how the search could indicate failure if the desired key did not exist in the array. Another mistake is to describe the process by describing the progress of a search of specimen data - rarely adequate because it misses many of the things that can happen with different data, and does not bring out the iterative nature of the process because the list is so short.

In part (c), it was necessary to indicate something about the workings of both techniques to gain both marks (preferably referring back to part (b)) - the bald (and common) “it looks at fewer records” shows little understanding. Not many appreciated the significance of the word “normally” - a linear search is actually much faster than a binary chop in finding a key such as “aardvark”!

Q15.

This question divided the candidates into two groups, those who could answer it and those who could not - an encouraging number scored nearly full marks, but an alarming number scored close to zero.

For part (a), although the while (TRUE) construction is standard terminology for an infinite loop few realised it, instead many tried wrongly to relate it to the state of the maintain variable, presumably because it was the only Boolean in sight.

Most candidates understood the data types for (b) - not many gave the correct reason for n being an integer (it is used as an array subscript), but many commented that it was only required to take the values 0 to 6 and so didn’t need to be anything else, perfectly valid reasoning. Many students could not even attempt the trace table, but many perfect attempts were seen, conversely some students made marking difficult by making elementary arithmetic errors early on which had to be laboriously followed through to give appropriate credit.

For the last part, which should have demonstrated ability to interpret the table in terms of the original problem, many elaborate flights of logic appeared, which were rewarded provided they were plausible and argued from the student’s trace table.

Q16.

This question was answered well by most candidates although there were some who had clearly never heard of a linked list

In part (a) most candidates were able to give the correct pointer values. The commonest error was to give positions in the alphabetical list.

In part (b) the values of start and free storage were usually correct.

Part (c) gave candidates the opportunity to explain what had to be done to add an item to

the list, without requiring re-call of an algorithm. Most candidates were able to explain some of the steps needed and good candidates gained all the marks. Weak candidates often placed the new item in the freestorage pointer instead of the array and forgot to increment freestorage. The method used to find the position of “monkey” in the list was rarely described in detail. Some candidates wanted to sort the array when the new item was added.

Q17.

This question deals with a standard data structure that should have been familiar to all candidates. A clear, labelled diagram was required for part (a). A collection of empty boxes, linked by arrows, with no labels scored zero. Linear and circular queues were both acceptable.

In part (b) candidates tended to score very well or very badly. Many candidates were obviously repeating an algorithm they had learned but others seemed to be inventing the algorithm in the examination. Most attempted to check for a full queue although not always in a sensible way. Some candidates failed to stop if the queue was full but proceeded to insert the data regardless. A fairly common approach was to make space by deleting an item. It was common to see assignment of data to the pointer rather than the queue location. Assignment was sometimes written the wrong way round so that the new data would be overwritten with the contents of the empty queue location or the pointer.

The presentation of the algorithm often left a great deal to be desired. If candidates make a large number of alterations then it is in their best interests to write out the final version neatly. Indenting correctly makes the algorithm more readable. Where the candidate uses identifiers the purpose of these should be explained.

Q18.

Many candidates grasped that the queue should be emptied into the stack, element by element, removing from the front of the queue and adding to the top of the stack. Candidates who answered by diagram lost marks if they did not clearly indicate that removal was from the front of the queue and insertion was at the top of the stack. Candidates then went on to state that the stack was emptied into the queue by popping elements in turn from the top of the stack. However, several candidates lost a mark by failing to reference the queue as the destination for the popped elements.