Section B

9. (a) *Award marks as follows, **up to [6 marks max]**.*
*Award **[1 mark]** for looping through the database and accessing all records.*
*Award **[1 mark]** for correct calculation of date difference (eg = today – paymentDate **OR** paymentDate – today).*
*Award **[1 mark]** for each list correctly compiled, **x3** (correct conditional statements according to date difference used).*
***Note:** Accept date difference not calculated/stated but assumed as "today – paymentDate" **OR** "paymentDate – today".*
*Award **[1 mark]** for successive if/else but wrong conditions.*
*Award **[1 mark]** for flagging correct records for deletion (do not accept deleting the records).*

Example:

```
set CURRDATE to current date (as a day number)
set LIST1, LIST2 and LIST3 to empty
loop through all CUSTREC in DATABASE
     DUEPERIOD = CURRDATE - CUSTREC.PAYMENTDATE
     if DUEPERIOD > 30 then
          add CUSTREC to LIST3
          flag CUSTREC to delete
     else if DUEPERIOD > 14 then
          add CUSTREC to LIST2
     else if  DUEPERIOD > - 30 then
          add CUSTREC to LIST1
     end if
end loop
```

[6]

***Note:** If candidates give their answer in flowchart form then credit them using the same marking points.*

(b) *Award up to **[4 marks max]**.*
(Using a mail merge facility);
Template for each type of reminder created in the word processor;
Lists created with customer ID;
Linked to customer details in database;
Appropriate details merged/inserted into template; **[4]**

(c) *Award **[1 mark]** for a consequence of data loss to customers and **[1 mark]** for a consequence of data loss to the insurance company.*

Example answer:
Customers would not be reminded when they needed to pay and some may overlook payment, hence not be insured;
The company could lose customers/ruin reputation; **[2]**

(d) *Award marks as follows up to **[3 marks max]**.*
*Award **[1 mark]** for a suitable measure and **[2 marks]** for a description related to the insurance company.*

Example answers:
Mirror system;
All changes to the records made on two systems;
If one fails then the other holds all current data;

Off site backup;
Snapshots/backups made on a regular basis;
In the case of failure a dated/time stamped copy exists and the state up until then can be used to restore customer records; **[3]**

10. (a) The OSI is a standardized system/model for network connection;
Consists of (7) layers;
Each dealing with specific parts of network communication;
For example, the physical layer which defines the physical connection; **[3]**

Note: *Award **[1 mark]** for the purpose of any of the 7 layers.*
*If candidate lists all 7 layers with no specific example award **[2 marks]** and a further **[1 mark]** if the purpose of at least one layer is given.*

(b) *Award up to **[3 marks max]**.*
Protocols are a set of rules;
To facilitate a process being carried out correctly;
(Used in each layer to ensure communication;)
For example (in the physical layer) the protocols could define the methods for opening and closing communication; **[3]**
Note: *Do not accept examples which are not related to networks.*

(c) *Award up to **[2 marks max]**.*
Name/ID;
Whether or not they are already a client;
If not a client, further details needed as input; **[2]**

(d) Two dimensional array;
With one column for each lawyer;
And one row per time slot; **[3]**
Note: *Accept column or row for lawyer and vice versa for time slot.*

(e) *Award up to **[5 marks max]**. Accept answers given as an algorithm.*
Loading ‘today’ page (and from now onwards or accept “start with tomorrow”);
If existing client, search appropriate lawyer/column only;
If not existing client, search time/row then lawyer/column;
Then allocate space if available;
If no space allocated, load next page and repeat until space found;
Then add client details to space; **[5]**

11. (a) *Award **[1 mark]** for data, **[1 mark]** for pointers, **[1 mark]** for order.*
Example:
Each node would hold the data for one plane (ID, place, time due, time expected, landed);
Head pointer points to the first in the list;
Each subsequent pointer points to the next in the list and last node has null pointer; **[3]**

(b) *Award **[1 mark]** for calculating hours.*
*Award **[1 mark]** for calculating minutes.*
*Award **[1 mark]** for input and output/return.*

Example 1:

```
input CTIME // time held in the collection in minutes
     HOURS = CTIME div 60
     MINUTES = CTIME mod 60
output HOURS, MINUTES // time to be displayed on the screen
```

Example 2:

```
input CTIME // time held in the collection in minutes
HOURS = 0
MINUTES = CTIME
WHILE MINUTES>59
     MINUTES=MINUTES-60
     HOURS=HOURS+1
ENDWHILE
output HOURS, MINUTES // time to be displayed on the screen
```

Example 3:

```
Format24 (CTIME)
// method accepts time held in the collection in minutes
     HOURS = CTIME div 60
     MINUTES = CTIME mod 60
     return  HOURS + ":" + MINUTES
     // returns time to be displayed on the screen
end Format24
```

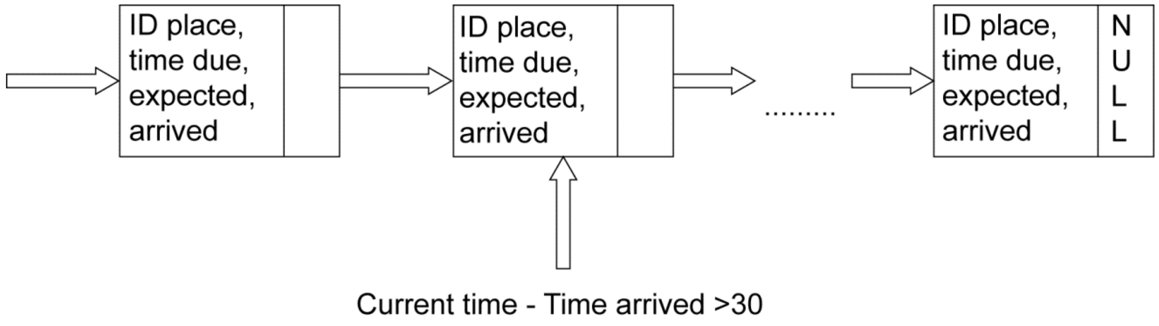
[3]

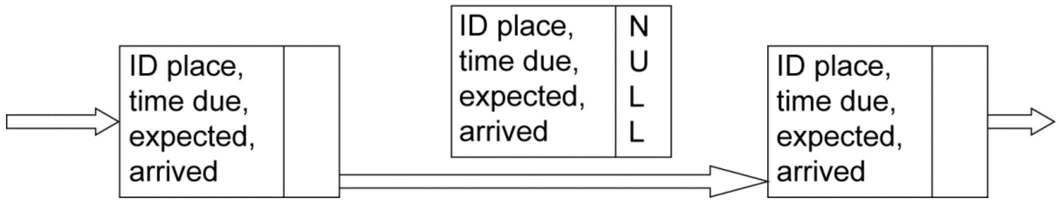
(c) *Award marks as follows, up to* ***[4 marks max]****.*
Award ***[1 mark]*** *for a diagram and explanation showing access to each plane via pointers;*
Award ***[1 mark]*** *for comparison of current time with time arrived;*
Award ***[1 mark]*** *for correct change of pointer from plane deleted;*
Award ***[1 mark]*** *for correct change of pointer to next plane;*

Note: *The plane to be deleted could be at the beginning of the list* ***OR*** *at the end of the list* ***OR*** *in the middle of the list; award third and fourth mark (change of pointers) depending on the position of the node shown in the candidates' diagram/explanation.*

For example:

```
PLANES accessed sequentially via pointers;
PLANE.ARRIVED checked against current time;
    if > 30 minutes;
        if pointer is head pointer;
            move head pointer to point to next PLANE;
        else if plane is last in list previous pointer points to NULL;
        else previous pointer changed to subsequent plane;
        pointer of deleted plane null;
```


ID place, time due, expected, arrived
ID place, time due, expected, arrived
....
ID place, time due, expected, arrived
NULL
Current time - Time arrived >30


ID place, time due, expected, arrived
ID place, time due, expected, arrived
NULL
ID place, time due, expected, arrived

[4]

(d) *Award up to* ***[5 marks max]****.*
A queue would hold the elements in order of arrival;
And enqueue correctly to the end as required;

Dequeue would take planes from the top of the screen;
Which is not wanted as they arrive at different times;

Elements in a linked list could be removed from any position in the list;
Hence a linked list is better;

Searching for ID to amend will be equivalent; **[5]**