**HL Unit 5** **– Abstract Data Structures**  
Quiz 1 – Linked Lists

|  |  |  |  |
| --- | --- | --- | --- |
| **Question 1** | | | |
| Objectives: | 5.1.12, 5.1.13 | Exam Reference: | May-14 6 |

Consider the following linked list which is maintained in alphabetical order.  
  
With the aid of diagrams, explain how the node  
  
  
would be inserted into the linked list. [3]

Initially compare with node pointed to by the head;

(If not correct place) move through list using pointers until correct alphabetical position

is found;

Adjust pointers accordingly;

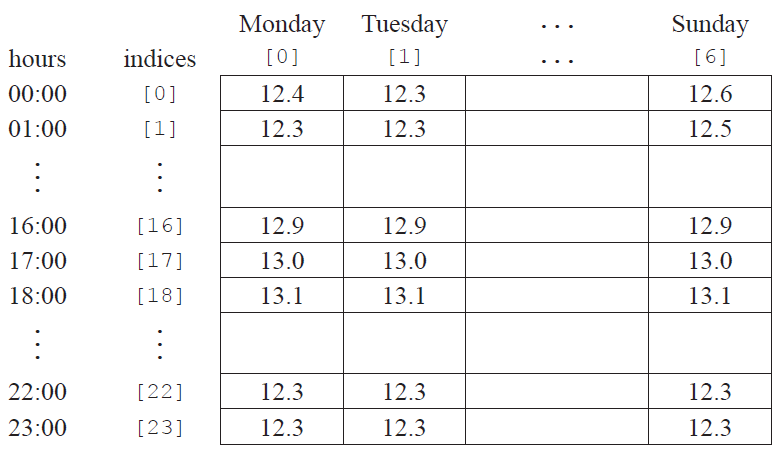
*(Drawings are acceptable, but award marks only if they clearly show how pointers are*

*correctly rearranged, following the three guidelines above.)*

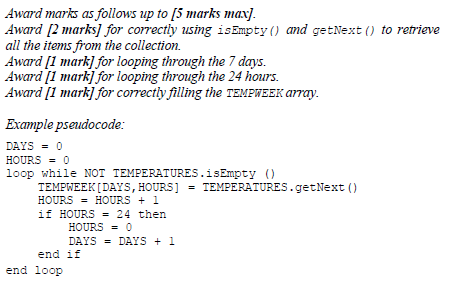
|  |  |  |  |
| --- | --- | --- | --- |
| **Question 2** | | | |
| Objectives: | 5.1.11 | Exam Reference: | Nov-14 11 |

The temperature (in °C) of a lake was recorded every hour, every day, for one week. As each reading was taken, it was added sequentially to the collection TEMPERATURES, which is stored permanently.

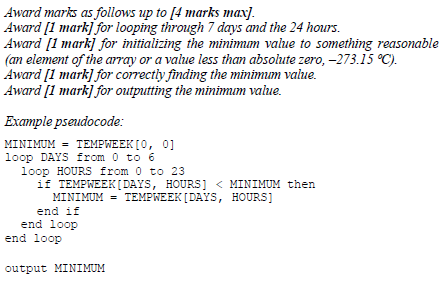
At the end of the week this data was read into a two-dimensional array named TEMPWEEK as shown below.



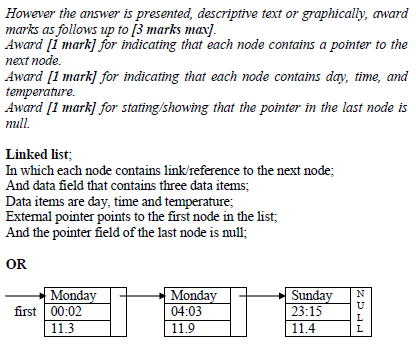
1. Construct the algorithm that will read the data from the collection into the array. You can use the collection functions TEMPERATURES.getNext() and TEMPERATURES.isEmpty(). [5]



1. Using the array TEMPWEEK, construct an algorithm to determine and output the minimum   
   temperature for the week. [4]



1. If the temperature is less than 12.0°C then the day, time and temperature are also placed in  
   a separate data structure.
2. Describe a dynamic data structure that might be used to hold this data. You may use a   
   labelled diagram. [3]



1. Using this dynamic structure suggest how the number of days when the temperature of   
   the lake was below 12.0°C can be found. [3]

*Award* ***[1 mark]*** *for each step identified up to* ***[3 marks max]****.*

Set counter to zero (0);

Start from the beginning of the list;

While the end of list is not reached;

Increase counter by 1;

Follow the pointers/links;

|  |  |  |  |
| --- | --- | --- | --- |
| **Question 3** | | | |
| Objectives: | 5.1.12, 5.1.13 | Exam Reference: | May-15 11 |

In a small airport, the details of all flights due to arrive on a particular day are held in a  
collection, FLIGHTS. Each object in the collection contains the following information:  
  
ID: unique flight number

PLACE: where the plane is coming from

DUE: the time it is scheduled to arrive

EXPECTED: the time it is expected to arrive (only if it is early or if it is delayed)

ARRIVED: the time of actual arrival.

EXPECTED and ARRIVED are blank at the beginning of the day and the collection is sorted in

order of DUE.

A screen in the airport can display information on 20 planes at a time, which are held in a

linked list.

1. Describe the features of a linked list of 20 planes that have the above information. [3]

*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);

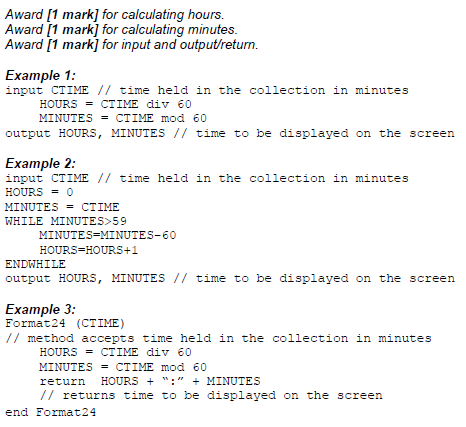
Head pointer points to the first in the list;

Each subsequent pointer points to the next in the list and last node has null pointer;

All times are stored in the collection as the number of minutes since midnight. However, they  
 are displayed on the screen in 24-hour format (for example, 10:58 is stored in the collection  
 as 658).

1. Construct an algorithm to convert the times held in the collection into hours and

minutes needed for the 24-hour format displayed on the screen. [3]



If a plane arrived more than 30 minutes ago it is removed from the linked list and the next  
 one in the collection is added to the end of the list.

1. With the aid of a diagram, explain how a plane which arrived more than 30 minutes ago

could be removed from the linked list. [4]

*Award marks as follows, up to* ***[4 marks max]****.*

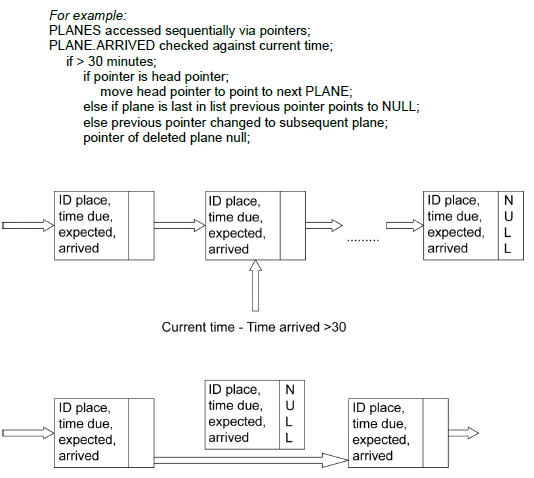
*Award* ***[1 mark]*** *for a diagram and explanation showing access to each plane via pointers;*

*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.*



1. For the application described above, compare the use of a linked list with the use of a

queue of objects. [5]

*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;