HL Unit 5 – Abstract Data Structures

Quiz 4 – Linked Lists

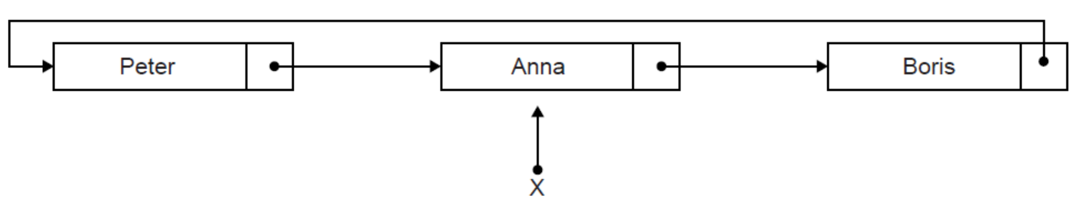
Question 1			
Objectives:	5.1.11	Exam Reference:	May-17 9

Identify the components of a node in a doubly linked list. [3]

Data;
A pointer/reference to the previous node;
A pointer/reference to the next node;

Question 2			
Objectives:	5.15.6, 5.1.11, 5.1.12, 5.1.13, 5.1.19	Exam Reference:	Nov-16 11

1. The diagram shows a list of names held in a circular linked list. The end of the list is pointed to by an external pointer, X.



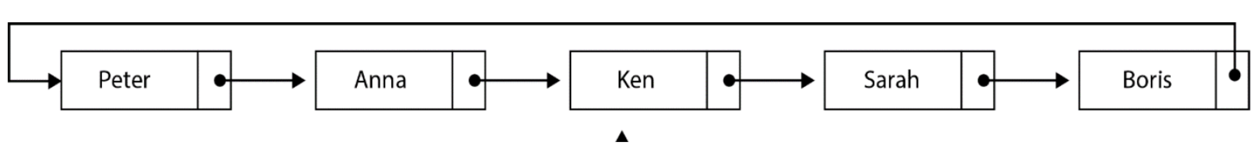
(a) State the first name in this circular list. [1]

Boris;

Two operations are performed on the list in the following order:
1. A node containing the name Sarah is inserted at the beginning of the list.
2. A node containing the name Ken is inserted at the end of the list.

(b) Sketch a diagram showing the resulting circular linked list. [3]

*Award up to **[3 max]**.*
For the diagram showing all nodes and links;
Ken inserted after Anna AND Sarah placed after Ken;
Node containing Ken is pointed to by X/Ken is currently at the end of the list;



(c) Describe how the number of names held in this list could be determined. [4]

Use a variable (counter) to keep track of/increment the number of nodes;
Use a temporary pointer;
Follow the pointers from the beginning of the list/from the node pointed to by pointer `X.next`;
Until the pointer to the end of the list (pointer X) is encountered;
Note: *Accept methods that start from the end of the list (X).*

(d) Explain how a stack could be used to output, in reverse order, all names held in the linked list. [4]

Traverse the list from beginning to end;
Pushing each data value from the list onto the stack;
While stack is not empty;
Popping an element from the stack and output the stack element;

(e) Compare the use of static and dynamic data structures. [3]

Static data structure has a predetermined number of elements but number of elements in dynamic data structure does not have to be defined in advance;
Static data structure has limited size, the amount of memory available is the only limit in size of dynamic data structure, size varies;
In static data structure elements can be directly accessed, in a dynamic data structure access is sequential (which is slower);

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).

(a) 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]

*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
```

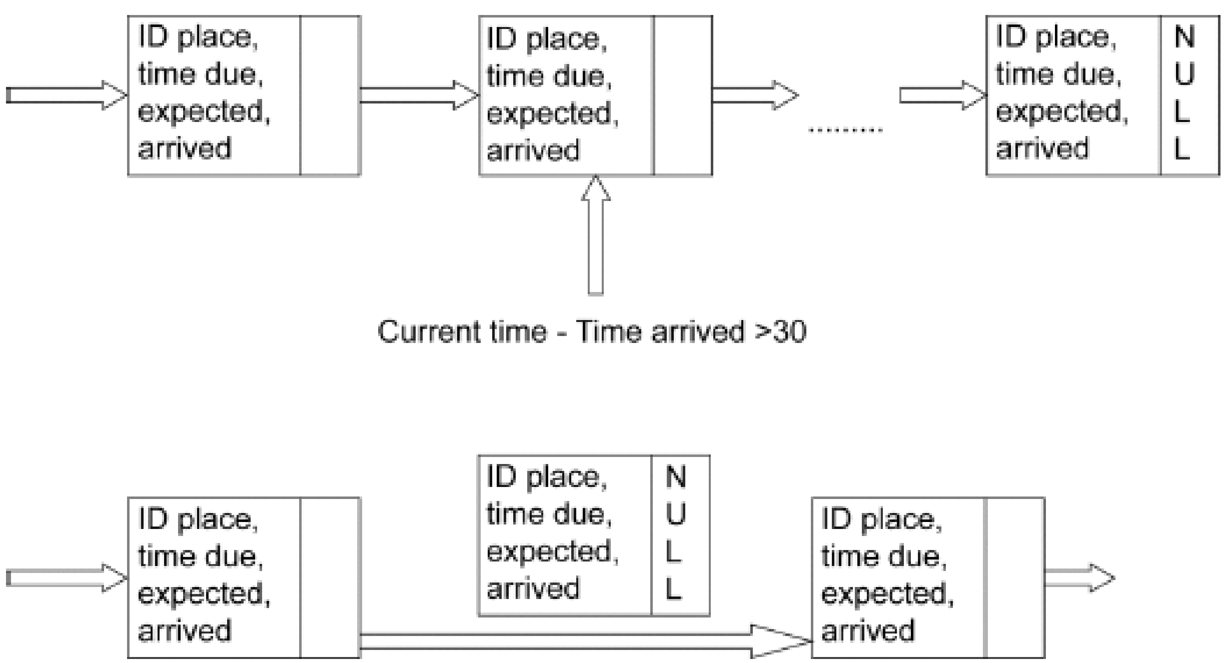
If a plane arrived more than 30 minutes ago it is removed from the linked list and the next one in the collection is added to the end of the list.

(b) 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.*

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;



(c) 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;