**HL Unit 5** **– Abstract Data Structures**  
Quiz 2 – 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;

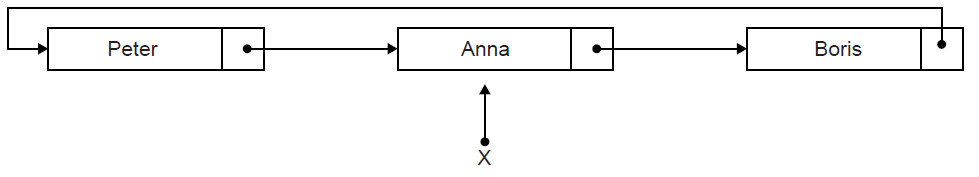
A pointer/reference to the previous node;

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.



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

Boris;

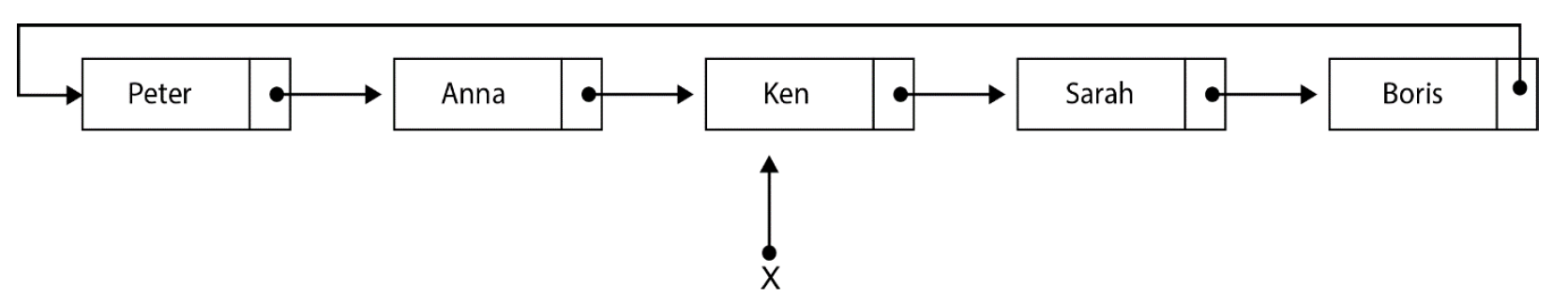
Two operations are performed on the list in the following order:  
 1. A node containing the name Sarah is inserted at the beginning of the list.  
 2. A node containing the name Ken is inserted at the end of the list.

1. 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;*

1. 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).*

1. 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;

1. 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);

|  |  |  |  |
| --- | --- | --- | --- |
| **Question 3** | | | |
| Objectives: | 5.1.12, 5.1.13 | Exam Reference: | Nov-17 13 |

(a) Describe the features of a dynamic data structure. [2]

*Award up to* ***[2 max]****.*

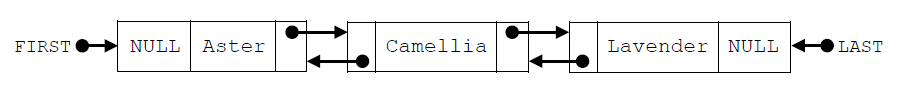
Each node contains data and also a link to other nodes;

Links between nodes are implemented by pointers (a pointer references a

location in memory or holds a memory address);

List size is not fixed / predetermined;

Consider the following doubly linked list which holds the names of flowers in  
 alphabetical order.



(b) Explain how “Primrose” could be inserted into this doubly linked list. You should   
 draw a labelled diagram in your answer. [6]

*Award up to* ***[6 max]*** *as follows. (There are 7 marking points)*

***[1]*** *create new node;*

***[1]*** *instantiation of values and pointers in new node;*

***[1]*** *state where the search starts from;*

***[1]*** *how to detect position for insertion;*

***[1]*** *update pointers in new node;*

***[1]*** *update pointers from the node at the insertion point, to the new node;*

***[1]*** *update external pointers;*

***Remark****: Some answers may just use illustrations alone, or very minimal explanations: see*

*note below;*

Create a new node (with pointer NEWNODE) with data field Primrose and two

pointer fields (next and previous), to be inserted;

Perform a linear search, either from the beginning or end of the list (using

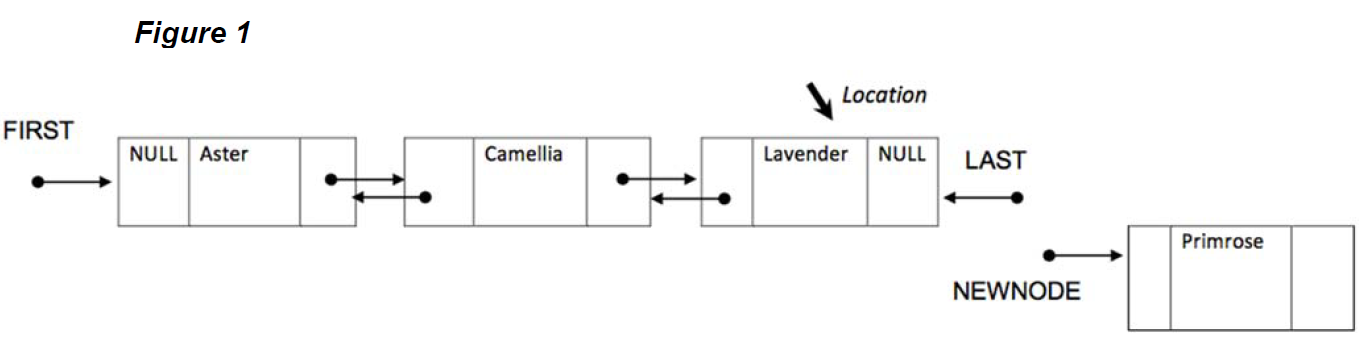
pointers FIRST and LAST, on the alphabetically order list;

The location/position of insertion, is found by comparing nodes (Primrose to be

inserted after Lavender, LOCATION points to Lavender) (***Accept*** *any description*

*to that effect*);

*(At the end of this phase, the situation looks as in* ***Figure 1****)*



Then, continue by setting the “next” field/pointer in the newly created node to

NULL;

Set the “previous” pointer in the newly created node to the current LAST / to point

to Lavender/ to point to the node detected by LOCATION;

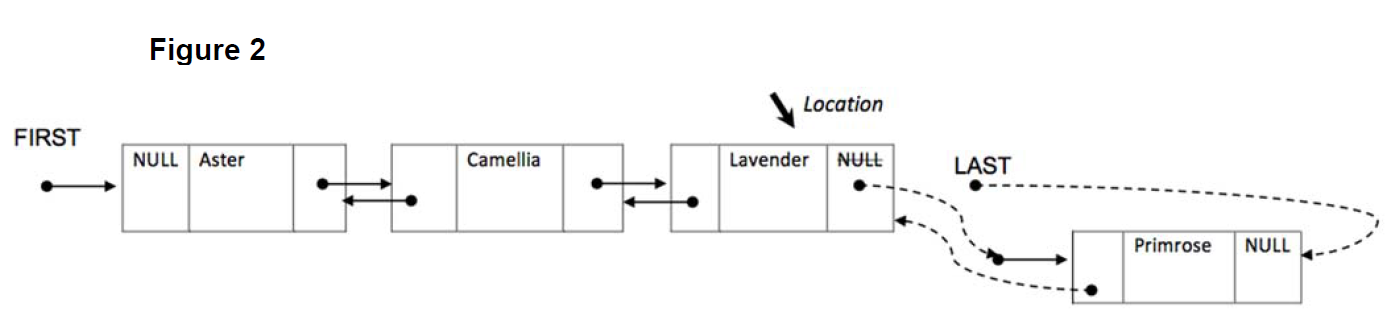
Change/Set/Update the Lavender’s “next” pointer to point to the new node / to

link with the NEWNODE pointer (delete NULL in the field and link to the existing

NEWNODE pointer);

Update the LAST pointer to point to the newly created node;

Eventually the final doubly linked list looks like this (**Figure 2**);



Consider the two stacks: FLOWERS and FRUITS.

***Note:*** *Award* ***[4 max]*** *for responses that return one or more drawings without any*

*explanation at all, for evidence of these features:*

***[1]*** *Evidence of creation of an initial new node for Primrose out of the list;*

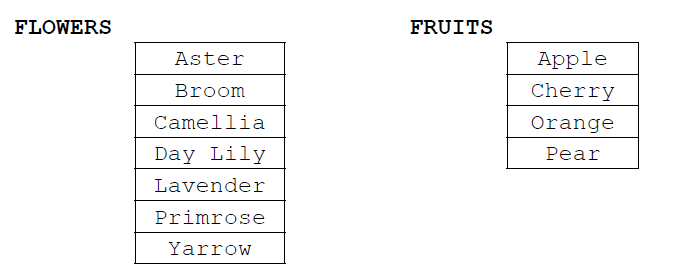
***[1]*** *The order of nodes Aster/Camellia/Lavender/Primrose is eventually correct;*

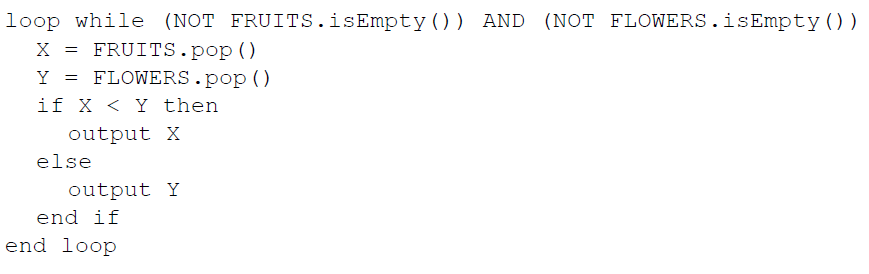
***[1]*** *The two unidirectional links between Lavender and Primrose are (eventually)*

*correctly displayed, from-to the appropriate fields;*

***[1]*** *LAST points correctly to the appropriate field in the new node Primrose,* ***and***

*NULL fills the last field of the new node;*



1. Show the output produced by the following algorithm. [4]

*Award* ***[1]*** *for each one in the correct order.*

Apple;

Broom;

Camellia;

Day Lily;

***Note:*** *Solution for the Spanish version (in this order):*

Aster; Camelia; Lavanda; Lirio;

A third stack, FLOFRU, is needed. It should contain all the data from FLOWERS and FRUITS and will store it as shown below

1. Describe how the FLOFRU stack could be created. [3]

*Award marks as follows up to* ***[3 max]****.*

***Example answer 1***

Create an empty stack (FLOFRU);

**pop** all elements from FRUITS and **push** them onto FLOFRU;

Then **pop** all elements from FLOWERS and **push** them onto FLOFRU;

***Example answer 2***

Create an empty stack (FLOFRU);

While FRUITS is not empty

**pop** an element from FRUITS and **push** it onto FLOFRU;

While FLOWERS is not empty

**pop** an element from FLOWERS and **push** it onto FLOFRU;

***Note:*** *Award* ***[2 max]*** *for generic descriptions that do not use appropriate*

*terminology on data structures and their operations.*