NO.

DATE 2/8/2023

* Recall:
  - ADT List - A collection of 0 or more elements.
    - Implementations/Representations
      - Array Implementation - Fixed-sized Array (ver. 1 & 2)
        - Dynamic-sized Array (ver. 3 & 4)
      - Linked-List Implementation - Singly-Linked
        - Doubly-Linked

CURSOR-BASED IMPLEMENTATION

* An implementation used by languages that do not support pointers.
  - (uses indexes of the array)
* A combination of array and linked list implementations
* In the cursor-based implementation, the list uses the array for its structure but manipulates the elements like the linked list way.

•COMPARISONS•

	•Array•	•Linked-List•
1.) Space	• Finite	• Depends on the memory of the PC (?)
2.) Execution	• can directly access elements w/ index	• Must traverse to reach any element
3.) Runningtimes	• insertLast is O(1). constant time	• insertLast is O(N)
	• insertFirst is O(N)	• insertfirst is O(1)

Sterling

NO.

DATE 2/8/2023

NOTE: With Linked-List implementations, the OS (operating system) handles the space. In cursor-based, we limit the space with the use of a Virtual Heap.

• Comparison Between LL & cursor based •

Linked - List	Cursor - based
• Pointer-to-Node	• int (index of the array)
• Pointer-to-Pointer--to-Node	• int *
• NULL	• -1

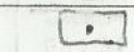
REVIEW

on: Linked-List Implementation

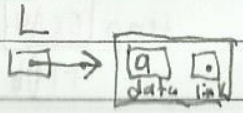
• Given Declaration:

```
typedef struct cell {
    char data;
    struct cell *link;
} ctype, *LIST;
LIST L;
```

• Illustration of an EMPTY List L:



• Illustration of a list w/ 1 element:



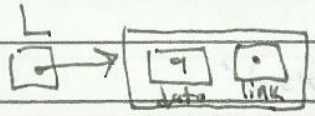
• Where in memory is the storage of the dynamically allocated cell taken from? IN THE HEAP

NO.

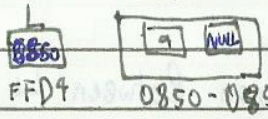
DATE 2/8/2023

Review Continuation

Conceptual View



Memory View



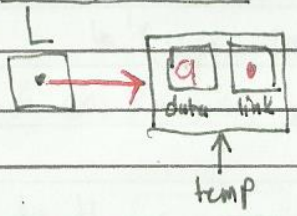
→ 9 bytes allocated

What is the value of L? 0850

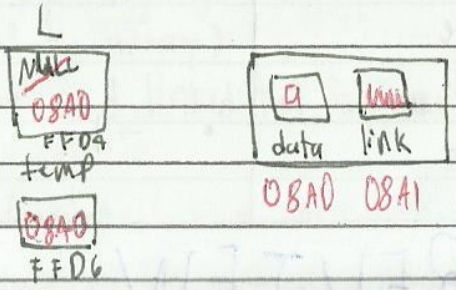
What is the value of L → link? NULL

// Inserting 'a' at the 1st position

Conceptual View

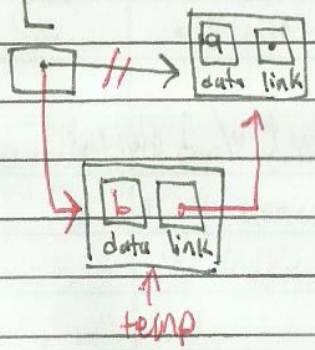


Memory view

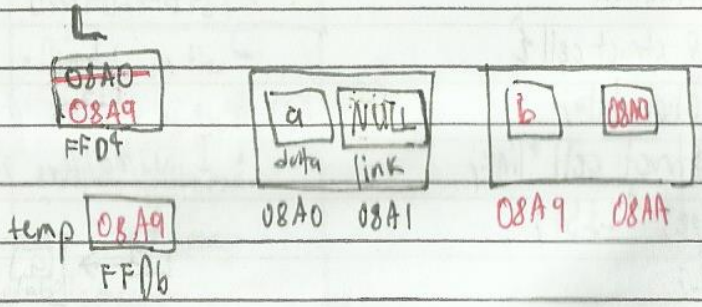


// Inserting 'b' at the 1st position

Conceptual View



Memory View

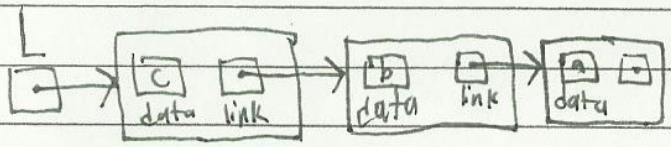


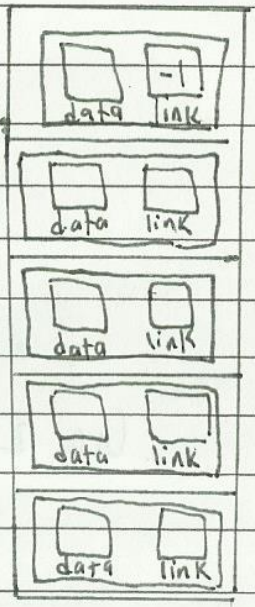
NO.

DATE 2/8/2023

What if? 🤔

Using Array Cells instead of dynamic Memory





Assumptions:

1.) Each array component (or cell) is a structure containing the data & the link field

2.) The next field is -1 if it is not pointing to any other cell. (That is, -1 is ≈ NULL).

3.) The order of array cel availability is from 0 to max - 1 (ex. max is 5)

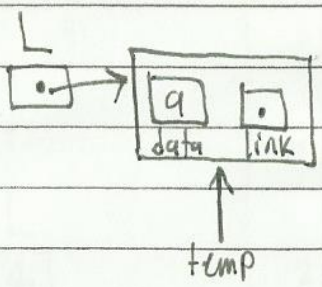
4.) The elements a, b, & c will be inserted in LIST L using Insert First () operation.

NO.

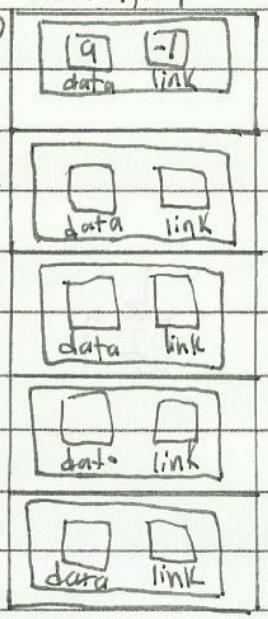
DATE 2/8/2023

Insert 'a' at the first position in L

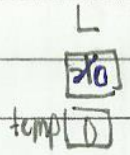
Linked List



arr Heap



List using Array Cells



★ Statement that will place 'a': arrHeap[temp].data = 'a';

L.L. equivalent(?): temp → data = 'a';

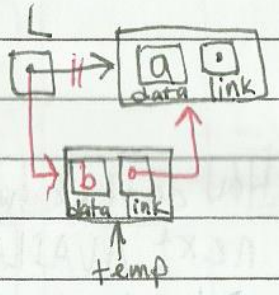
★ Statement that will set link to null: arrHeap.link = -1;

L.L. equivalent(?): temp → link = NULL;

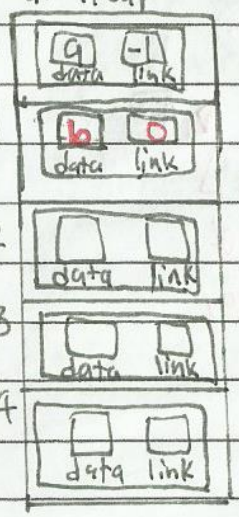
NOTE: arrHeap → is an array
arrHeap[temp]. → is a structure (uses a dot •)
arrHeap[temp].data → is a character

Insert 'b' at the first position in L

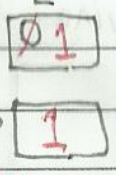
Linked List



arr Heap



List using Array Cells



assumption of next availability

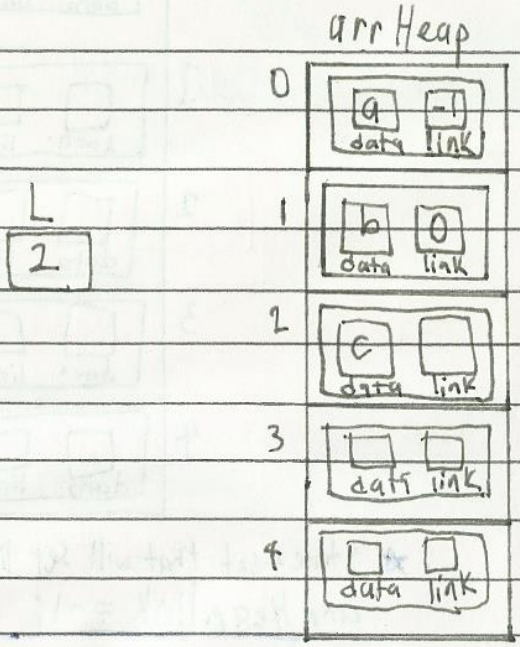
Sterling

NO.

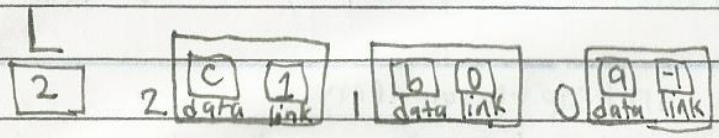
// my hands are in pain

DATE 2/8/2023

- Cursor-based Implementation uses ARRAY OF CELLS instead of dynamic memory cells.

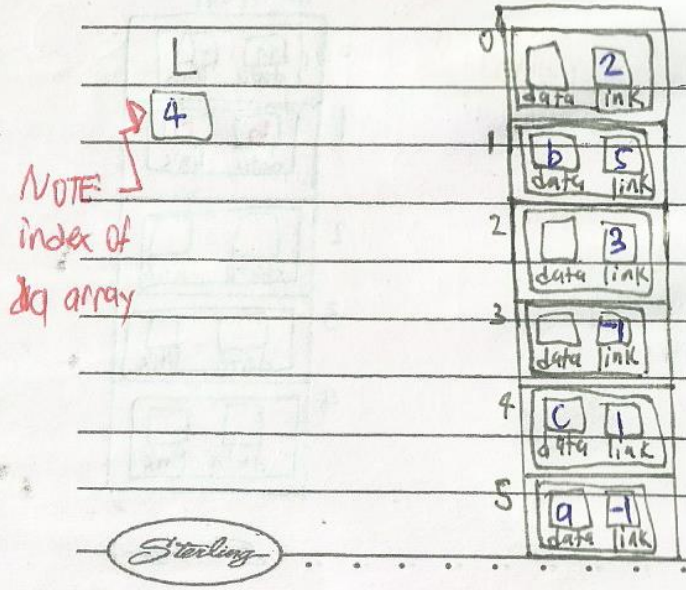


★ different View of List L =



- Suppose, array cell availability is NOT from 0 to max-1... but RANDOM.

★ Assumption: Elements are in random cells (L elements: c, b, a)



How do you know which is the next AVAILABLE cell?

→ Linking available spaces

→ having Avail

0 ★ NOTE: L index of arr array

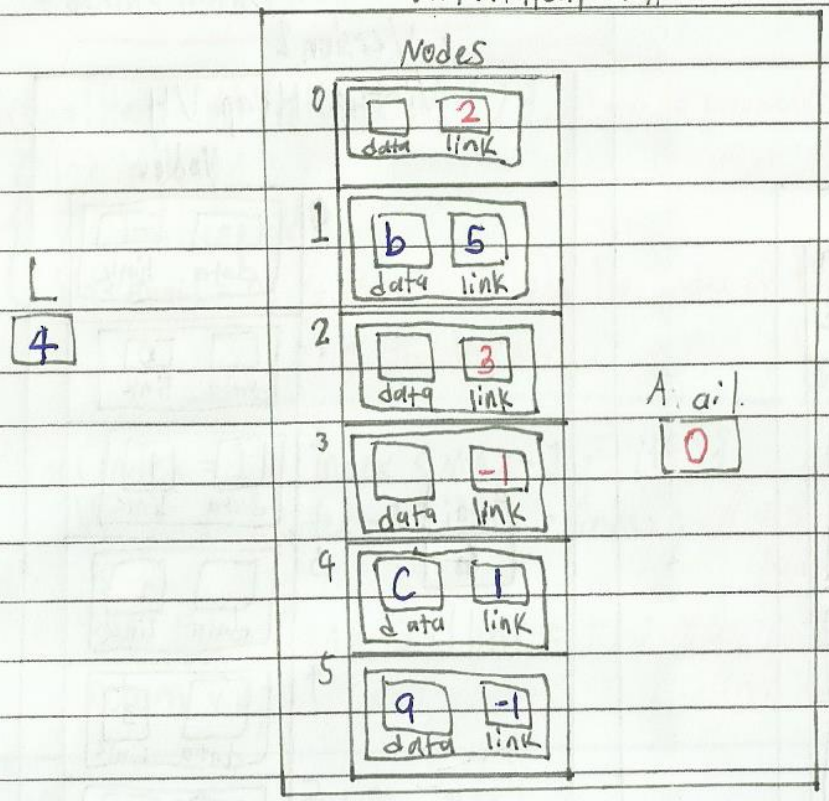
Sterling

NO.

DATE 2/8/2023

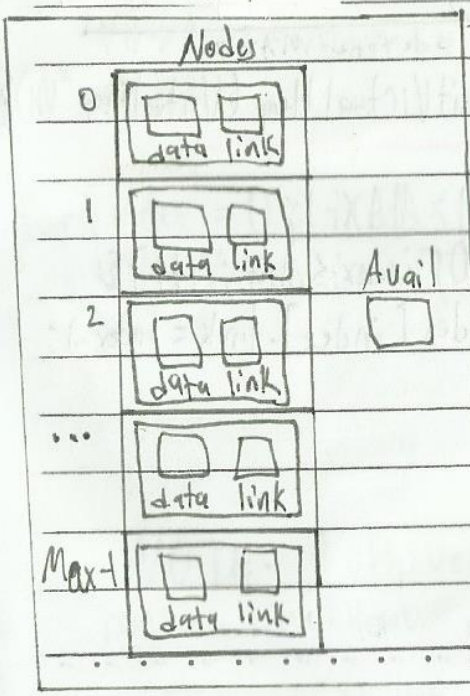
• Creating an imaginary HEAP •

Virtual Heap VH



Exercise: Write an appropriate definition of the following data types: based on dis illustration:

Virtual Heap VH



1.) Virtual Heap

```
#define MAX 10
typedef struct {
char data;
int link;
} Nodetype;
```

```
typedef struct {
Nodetype Nodes;
int Avail
} VirtualHeap;
```

2.) List

```
typedef int List;
```

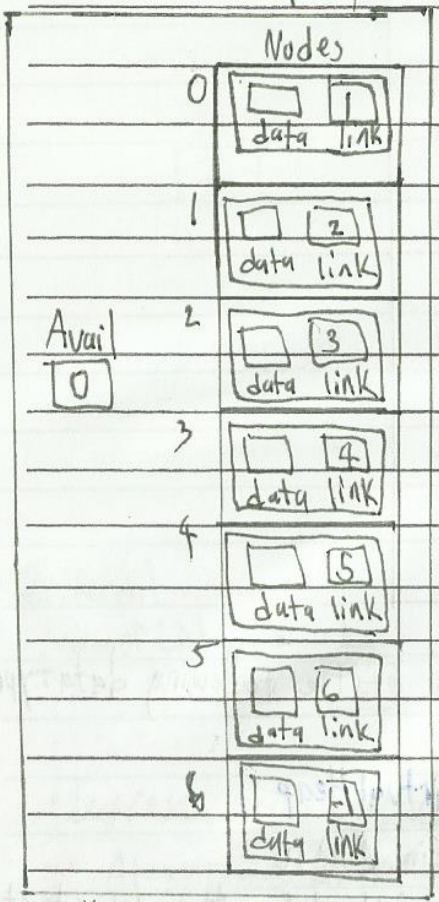
Sterling

NO.

DATE 2/8/2023

Exercise: initializing the VirtualHeap (2 versions)

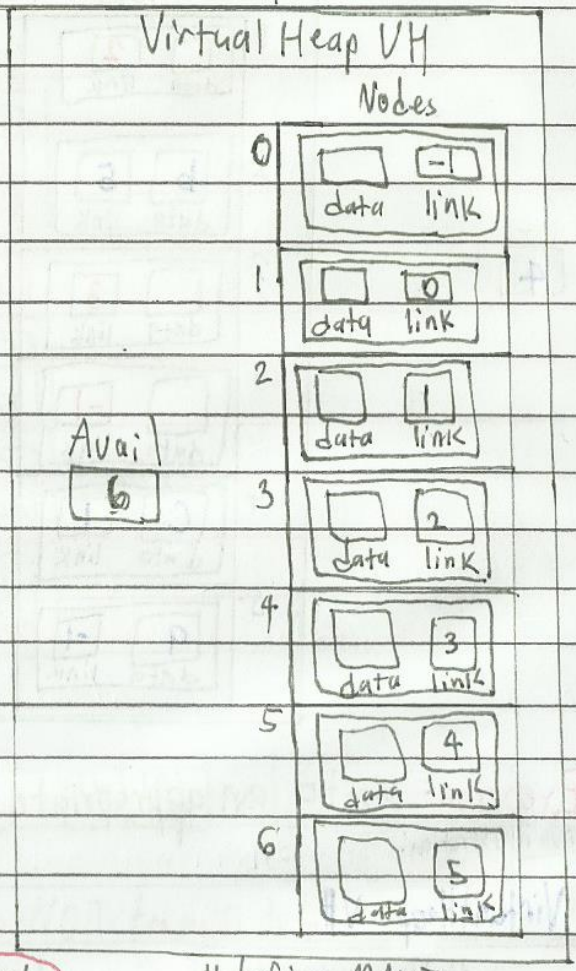
• Version 1



Code:

```
#define MAX 7
VirtualHeap initVirtualHeap() {
  VirtualHeap VH; int index;
  VH.Avail = 0;
for(index = 1; index < MAX-1; itt){
  VH.Nodes[index-1].link = index;
}
  VH.Nodes[MAX-1].link = -1;
  return VH;
}
```

• Version 2



Code:

```
#define MAX 7
void initVirtualHeap(VirtualHeap *VH){
  int index;
  VH->Avail = MAX-1;
  for(index = 0; index < MAX; itt){
    VH->Nodes[index].link = index-1;
  }
}
```

Sterling

Writers: How tf do i explain this

NO.

DATE 2/8/2023

CODE EXPLANATION

- Version 1 code:

```
#define MAX 7

VirtualHeap initVirtualHeap ()){   // No parameters because it will pass a garbage value, declare it as local variables instead.
    int index;                     // declare local variables
    VirtualHeap VH;
    VH.Avail = 0;                  // Avail will hold the index of the 1st available node
                                   // in the array.

    for(index = 1; index < MAX-1; i++){   // index will hold the link of next node (??)
        VH.Nodes[index-1].link = index;   // assigning the link to the next node/cell
    }
    VH.Nodes[MAX-1.]link = -1;     // assigning the LAST node's link to Null (-1).
    return VH;
```

- Version 2 code:

```
#define MAX 7

void initVirtualHeap (VirtualHeap *VH){
    int index;
    VH->Avail = MAX-1;             // Avail will hold the index of the 1st available
                                   // node.

    for(index = 0; index < MAX; i++){   // index holds the node # (?!??)
        VH->Nodes[index].link = index-1;   // assigning the link of each
                                           // node in the array to the
                                           // previous node (or to null if
                                           // index 0)
    }
}
```

Side NOTE:
VH-> can also be (VH*)
but not advisable

NOTE: Both versions have around the same patterns & running time. However, Version 2 is better in terms of consiceness of code.

Sterling

NO.

DATE 2/8/2023

★ 3 Virtual Heap Management Routines ★

1.) initializeVH() - also can be initVirtualHeap()
- Given a virtual heap, this function will link all nodes in the array

NOTE: The value -1 is equivalent to NULL & indicates the end of the list.

2.) allocSpace() - equivalent to malloc()
- this function will remove the first available node in the virtual heap & return the index of that node to the calling function.

NOTE: If -1 is returned, no more available space.

3.) deallocSpace() - equivalent to free().
- Given the index of a node, the function will put back the node in the list of available nodes in the virtual heap.

MOAR NOTES: In Linked-List, the management of the heap memory is handled by the compiler & Operating System. In cursor-based, the virtual heap is managed by the programmer through the virtual heap management Routines.

Sterling