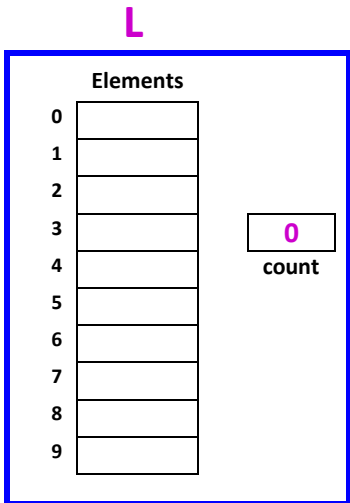
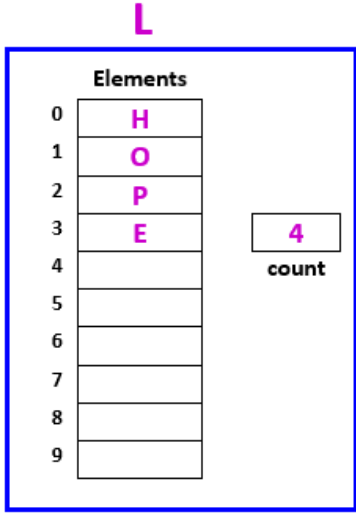


Array Implementations of List and Initialize Function

Array Implementation of a List of characters		
Definition and Declaration: <pre>#define SIZE 10 typedef struct { char elements[SIZE]; int count; }List; List L;</pre>	Figure 1: Empty list L. 	Figure 2: List L with 4 elements: 'H', 'O', 'P', and 'E'. 

Notes:

- 1) Figures 1 and 2 shows the **maximum capacity** of the lists.
- 2) The **count** field/member holds the actual number of elements in the list.
- 3) Figure 2 shows the characteristic of an array implementation of list, i.e., **"Elements are stored in contiguous cells of an array"**.

Exercise 1:

Function **initializeList()**. The function will initialize the list to be empty (See Figure 1). This can be done in two ways:

- a) Given the list as parameter, and setting count to 0. **Note:** A part of the list is modified, thus determine how the list should be passed as parameter.
- b) Set the count field to 0 of a locally declared list which is returned to the calling function. **Note:** Locally declared structures can be returned to the calling function.

Write the code of the two functions representing the two versions of **initializeList()**.

Exercise 2: Write the code of the following functions, based on the given definition of List above and definition of Boolean below:

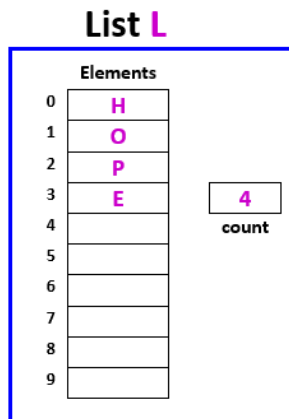
```
typedef enum {TRUE, FALSE } Boolean;
```

Note: Make the statements in the code general (works for all situations). Avoid statements such as: if the list is empty, do the following; else do the following

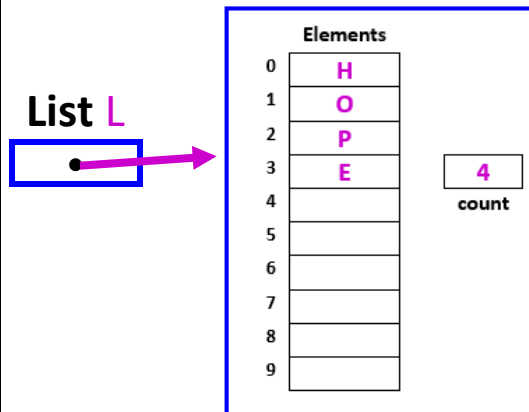
- a) Function **findElem()**. The function will return the index of the 1st occurrence of the given element in the given list. If the element is not found return -1.
- b) Function **insertFirst()**. The function will insert a new given element in the given list at the 1st position if there is still space in the list. This is done by moving existing elements 1 index higher to make room for the new element and updating necessary variables.
- c) Function **deleteElem()**. Given the element and the List, the function will delete the given element if found and return 1 to the calling function; otherwise return 0. Deletion of the element in the list is done by moving succeeding elements 1 index lower than their original position and updating necessary variables.

Variations of Array implementation of List (Illustrations with 4 elements)

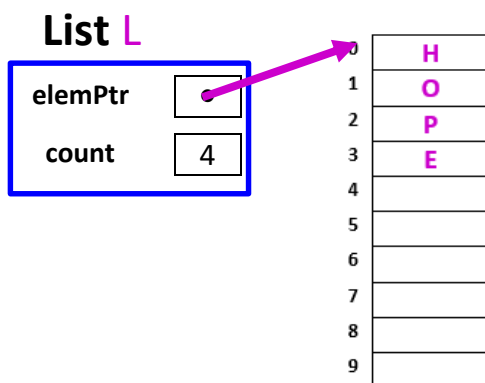
Version 1: List is a structure containing an array and variable count.



Version 2: List is a pointer to a structure containing an array and variable count

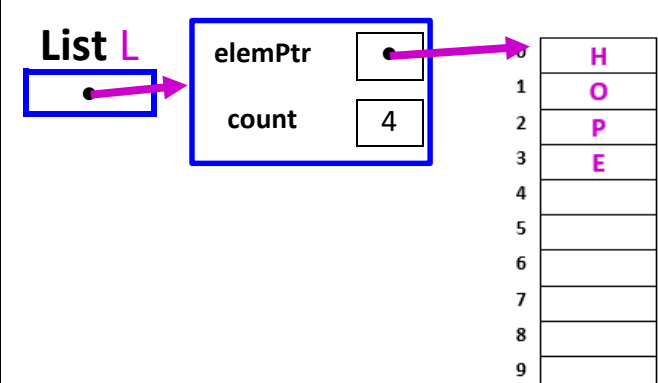


Version 3: List is a structure containing variable count and a pointer the first element of a dynamically allocated array.



Note: The array can grow or shrink using realloc().

Version 4: List is a pointer to a structure containing variable count and a pointer the first element of a dynamically allocated array.



Note: The array can grow or shrink using realloc().

Exercise 3: For Versions 2, 3, and 4 ONLY.

- 1) Write an appropriate definition of data type List and Declaration of variable L for each of the variation of the array implementation of List. In addition, define as a macro, SIZE will represents the size of the array.
- 2) For each of the 3 versions or variations, write the code of the function initializeList().