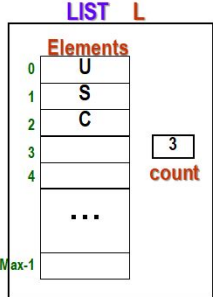
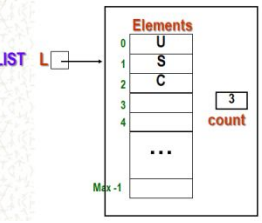
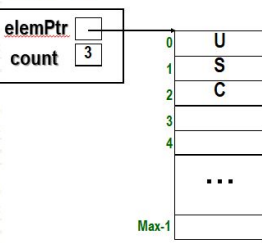


A. Define each of the following in one (1) sentence.

1) Abstract data type (ADT)
- An abstract data type is defined by its behavior from the point of view of a user, of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations.
2) List
- a series of names or items written or printed together in a meaningful grouping or sequence so as to constitute a record.
3) ADT List
- an ordered collection of items of some element type E. it just means that each object has a <i>position</i> in the List, starting with position 0.

B. Complete the table.

4 Versions of the array implementations of ADT List	Illustrations of lists containing characters as elements.	Do the following: 1) Define Max as a macro using <code>#define</code> 2) Write a definition of datatype LIST and declaration of variable L. 3) Beside the declaration of L, comment the number of bytes allocated to L.	Write the code of function <code>initializeList()</code> . The function will create the dynamically allocated spaces (if they exist) and set the given list to be empty.
1) List is a structure containing an array and variable count		<pre>#define Max 10 typedef struct node { char Elements[Max]; int count; }LIST; LIST L; //14 bytes</pre>	<pre>LIST initializeList() { LIST T; T.count = 0; return T; }</pre>
2) List is a pointer to a structure containing an array and variable count		<pre>#define Max 10 typedef struct node { char Elements[Max]; int count; }*LIST; LIST L; // 8 bytes</pre>	<pre>LIST initializeList () { LIST *t = (LIST) malloc(sizeof(struct node)); if (t != NULL) { t->count = 0; } return t; }</pre>
3) List is a structure containing variable count and a pointer to a dynamic array		<pre>#define Max 10 typedef struct node { char *elemPtr; int count; } LIST; LIST L; //12bytes</pre>	<pre>LIST initializeList (LIST t) { if (t != NULL) { t->elemPtr = (char *)malloc(Max); if (t->elemPtr != NULL) { t->count = 0; } } return t; }</pre>

4) List is a pointer to a structure containing variable count and a pointer to a dynamic array		<pre>#define Max 10 typedef struct node { char *elemPtr; int count; } *LIST; LIST L; // 8 bytes</pre>	<pre>void initializeList (LIST *t) { *t = (LIST) malloc (sizeof(struct node)); if (*t != NULL) { (*t)->elemPtr = (char *)malloc(Max); (*t)->count = 0; } }</pre>
--	--	--	--

Define each of the following in one (1) sentence.

1) ADT Stack
- A stack is an abstract data type that serves as a collection of elements, with two main principal operations: push, which adds an element to the collection, and pop, which removes the most recently added element that was not yet removed.
2) ADT Queue
- A queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence.

C. Complete the table below by determining the equivalent names and description of the ADT list operations in ADT Stack and ADT Queue.

ADT List Operations Note: Position P starts with 0.	ADT Stack	ADT Queue
1) Insert(L, P, E) - Inserts an element E at position P in the given list L.	push (S, E) – inserts an element E at the top/end of the stack S.	enqueue (Q, E) – inserts an element E at the last/rear of the queue Q.
2) delete(L, P) - Deletes the element at position P of the given list L.	pop (S) – deletes the element at the top/end element of the stack S if it is not empty	dequeue (Q) – deletes the first/front element of the queue Q if it is not empty.
3) retrieve(L, P) – Retrieves and returns the element at position p of the given list L.	peek(S) – fetch the top/end element of the stack S if it is not empty.	peek (Q) – fetch the element at the first/front element of the queue Q if it is not empty.

D. Submission Details:

- File name : CIS2101_Ass_03_lastname
- Email to chetpena@gmail.com with **email subject:** CIS 2101 – Ass #3
- PLEASE follow instructions.