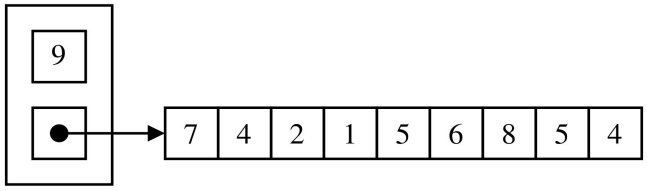
CS19001/CS19002 PROGRAMMING AND DATA STRUCTURES LABORATORY
Assignment No: 8
Last Date of Submission: 30–March–2015

In this assignment, you make an implementation of an ordered list of integers using dynamic memory. The list consists of a header storing two items: 1) The size of the list, and 2) A dynamically allocated array capable of storing *only* the elements in the list. In the picture below, the ordered list is (7,4,2,1,5,6,8,5,4). Its size is 9, and the dynamic array must be allocated sufficient memory to store *exactly* nine integers.



Part 1: Define a structure to store the size of the list and a pointer to the array of list elements.
Part 2: Write a function to return an empty list. For an empty list, the size is zero, and the array is NULL.
Part 3: Write a function that, given a list *L* and an integer *a*, appends *a* at the end of the current list *L*, and returns the modified list.
Part 4: Write a function that, given a list *L* and an integer *a*, prepends *a* at the beginning of the current list *L*, and returns the modified list.
Part 5: Write a function that, given a list *L*, deletes the last element of the list, and returns the modified list.
Part 6: Write a function that, given a list *L*, deletes the first element of the list, and returns the modified list.
Part 7: Write a function that, given a list *L* and an integer *a*, deletes all occurrences of *a* in *L*, and returns the modified list.

In Parts 3–6 (and possibly also in Part 7), you need to change the allocation size of the list array. One possibility is to use realloc(). Another possibility is malloc()-copy-free().

Write a main() function that tests the functions of Parts 2–7. See the sample output below.

Submit a single C file solving all the parts.

Sample Output

```
+++ init()          : ()
+++ append(9)       : (9)
+++ append(1)       : (9,1)
+++ append(2)       : (9,1,2)
+++ append(3)       : (9,1,2,3)
+++ append(6)       : (9,1,2,3,6)
+++ append(7)       : (9,1,2,3,6,7)
+++ append(8)       : (9,1,2,3,6,7,8)
+++ append(4)       : (9,1,2,3,6,7,8,4)
+++ append(2)       : (9,1,2,3,6,7,8,4,2)
+++ append(3)       : (9,1,2,3,6,7,8,4,2,3)
+++ prepend(9)      : (9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(2)      : (2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(4)      : (4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(8)      : (8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(2)      : (2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(5)      : (5,2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(9)      : (9,5,2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(8)      : (8,9,5,2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(9)      : (9,8,9,5,2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ prepend(1)      : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6,7,8,4,2,3)
+++ deletelast()    : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6,7,8,4,2)
+++ deletelast()    : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6,7,8,4)
+++ deletelast()    : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6,7,8)
+++ deletelast()    : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6,7)
+++ deletelast()    : (1,9,8,9,5,2,8,4,2,9,9,1,2,3,6)
+++ deletefirst()   : (9,8,9,5,2,8,4,2,9,9,1,2,3,6)
+++ deletefirst()   : (8,9,5,2,8,4,2,9,9,1,2,3,6)
+++ deletefirst()   : (9,5,2,8,4,2,9,9,1,2,3,6)
+++ deletefirst()   : (5,2,8,4,2,9,9,1,2,3,6)
+++ deletefirst()   : (2,8,4,2,9,9,1,2,3,6)
+++ deleteall(7)    : (2,8,4,2,9,9,1,2,3,6)
+++ deleteall(2)    : (8,4,9,9,1,3,6)
+++ deleteall(9)    : (8,4,1,3,6)
+++ deleteall(4)    : (8,1,3,6)
+++ deleteall(7)    : (8,1,3,6)
```