July 2025

CSE 106 - Assignment 1

CSE, BUET

In this assignment, you will be implementing List ADT (Abstract Data Type - Data types that are defined by their behaviors rather than their implementation). List ADT helps to store the same type of items sequentially. You will implement List ADT for storing integers from scratch.

Task

The task of this assignment is to implement the List ADT for integers. Suppose there is a list containing 3, 4 and 5 sequentially and the current position is 1, it will be shown as following:

[3 4| 5]

'|' (vertical bar) character indicates that the current position is 1. Empty list should be shown as following:

[.]

The functionalities you need to implement are shown in the following table (assume before every operation the list holds 3, 4, 5 and current position is 1):

Function Number	Function Name	Parameter	Return Value	List After Execution	Comment
1	Insert_cur(int item)	6	-	[3 4 6| 5]	Insert at current position
2	delete_cur()	-	4	[3 5|]	Delete the element at current position
3	append(int item)	1	-	[3 4| 5 1]	Append at the end of the list
4	size()	-	3	[3 4| 5]	Return the number of elements

5	prev(int n)	1	-	[3| 4 5]	Go to the n-th previous element (if there are less than n previous elements, go to the first element)
6	next(int n)	3	-	[3 4 5|]	Go to the n-th next element (if there are less than n next elements, go to the last element)
7	is_present(int n)	3	1	[3 4| 5]	If the integer is present in the list at least once return 1, otherwise return 0
8	clear()	-	-	[.]	Clear the list
9	delete_item(int item)	3	1	[4| 5]	Delete the item from the list if present (first appearance)
10	swap_ind(int ind1, int ind2)	0, 2	-	[5 4| 3]	Swap elements between two indices
11	search(int item)	5	2	[3 4 5|]	Search for an item and return its index (first appearance). If it is not present, return -1.
12	find(int ind)	2	5	[3 4 5|]	Find the element at the given index. If the index is out of range, return -1
13	update(int ind, int value)	0, 7	3	[7| 4 5]	Update the element at the given index position and return the previous element
14	trim()	-	5	[3 4|]	Delete the last element and return it
15	reverse()	-	-	[5 4| 3]	Reverse the entire list

You have to give two implementations of List ADT:

1. Array List

It is implemented using an array. To make it efficient, you have to double the size of the array, when the array is more than 50% full (e.g. if the size of the array is 8, you have to increase the size to 16 when the 5th element is inserted.) Again, you have to shrink it to half of the size, when it is less than 25% full (e.g. if the size of the array is 8, you have to shrink it to 4 when there are less than 2 elements [For size 0, keep capacity 2]. YOU MUST USE DYNAMIC MEMORY ALLOCATION.

2. Linked List

You have to implement a doubly linked list. To do that, you need to define a 'struct' for nodes.

You are provided with skeleton codes. See the comments in that skeleton code for better understanding.

Following table shows a sample input and output:

Sample Input	Sample Output
1 1 1 2 1 3 1 4 1 5 1 6 5 2 1 0 2 9 5 9 3 9 100 2 2 2 2 3 7 3 8 3 9 3 10 4	Insert 1 [1|] Insert 2 Capacity increased from 2 to 4 [1 2|] Insert 3 Capacity increased from 4 to 8 [1 2 3|] Insert 4 [1 2 3 4|] Insert 5 Capacity increased from 8 to 16 [1 2 3 4 5|] Insert 6 [1 2 3 4 5 6|] Prev 2 [1 2 3 4| 5 6] Insert 0 [1 2 3 4 0| 5 6]

7 8 7 11 3 11 10 1 3 6 10 5 3 8 1 12 1 50 1 34 11 5 11 34 12 5 12 -1 12 1 13 5 1 13 2 5 14 1 15 5 2 14 15 1 15 1 20 15	Delete current item [1 2 3 4 5| 6] 0 is deleted Delete 5 [1 2 3 4 6|] Delete 3 [1 2 4| 6] Delete 100 100 not found Delete current item Capacity decreased from 16 to 8 [1 2 6|] 4 is deleted Delete current item [1 2|] 6 is deleted Delete current item Capacity decreased from 8 to 4 [1|] 2 is deleted Delete current item Capacity decreased from 4 to 2 [.] 1 is deleted Append 7 [7|] Append 8 Capacity increased from 2 to 4 [7| 8] Append 9 Capacity increased from 4 to 8 [7| 8 9] Append 10 [7| 8 9 10] Size of the list is 4 8 is present 11 is not present

```
Append 11
Capacity increased from 8 to 16
[ 7| 8 9 10 11 ]

Swap index 1 and 3
[ 7| 10 9 8 11 ]

Next 10
[ 7 10 9 8 11| ]

Prev 3
[ 7 10| 9 8 11 ]

Clear list
[ . ]

Insert 12
[ 12| ]

Insert 50
Capacity increased from 2 to 4
[ 12 50| ]

Insert 34
Capacity increased from 4 to 8
[ 12 50 34| ]

Search 5
5 is not found

Search 12
[ 12| 50 34 ]
12 is found at 0

Find 5
5 is not a valid index

Find -1
-1 is not a valid index

Find 1
[ 12 50| 34 ]
50 is found at 1

Update element at 5
5 is not a valid index

Update element at 2
[ 12 50 5| ]
34 is updated by 5
```

	Trim [12 50|] 5 removed Insert 15 [12 50 15|] Prev 2 [12| 50 15] Trim [12| 50] 15 removed Reverse [50| 12] Insert 15 [50 15| 12] Insert 20 [50 15 20| 12] Reverse [12 20 15| 50] End

The sentences in red color are only applicable for arrayList implementation.

For this assignment, you have to use C/C++. Skeleton code, along with sample inputs and outputs, are also provided.

Marks Distribution

Tasks	**Marks**
ArrayList resize	15%
ArrayList functions (15 x 2)	30%
LinkedList functions (15 x 2)	30%
Proper memory allocation and deallocation	15%
Proper printing	10%
Total	100%

Submission Guidelines:

1) Create a new folder with your 7-digit student ID.
2) Copy your source files into the folder created in step 1.
3) Zip the folder (along with the source files). Make sure it has the ".zip" extension. No other extension will be accepted.
4) Upload the zip file into the designated submission link on Moodle.

Suppose your student ID is xx05xxx. If so, create a folder named "xx05xxx", put your source files in that folder, compress/zip the folder into xx05xxx.zip, and finally, upload xx05xxx.zip to Moodle

Please make sure to follow the guidelines. You might be penalized up to 10% of the total marks if you do not comply. Also make sure to upload the correct zip file (the file is not corrupted or is not the solution to a previous assignment you did). Please do not copy code from other students or any other sources. Copying from others results in up to -100% penalty. We expect honesty and integrity from you.