

Lab 03 Implementation of Circular Doubly Linked Lists in C language

Objectives:

- Learn to create the circular doubly linked lists.
- Implement node traversal in circular doubly linked lists.
- Implement other linked list operations such as node insertion and deletion.
- Learn to use linked lists in practical applications by solving classical Josephus Problem

Pre-Lab Reading

Circular Doubly Linked Lists

A circular doubly linked list or a circular two-way linked list is a more complex type of linked list which contains a pointer to the next as well as the previous node in the sequence. The difference between a doubly linked and a circular doubly linked list is same as that exists between a singly linked list and a circular linked list. The circular doubly linked list does not contain NULL in the previous field of the first node and the next field of the last node. Rather, the next field of the last node stores the address of the first node of the list, i.e., START . Similarly, the previous field of the first field stores the address of the last node. A circular doubly linked list is shown in *Figure 1* below:

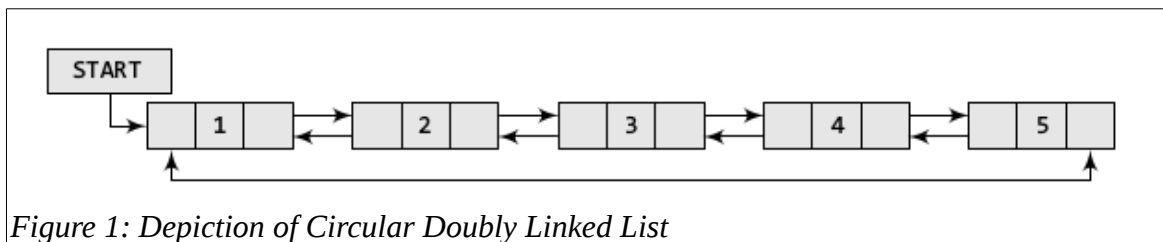


Figure 1: Depiction of Circular Doubly Linked List

Since a circular doubly linked list contains three parts in its structure, it calls for more space per node and more expensive basic operations. However, a circular doubly linked list provides the ease to manipulate the elements of the list as it maintains pointers to nodes in both the directions (forward and backward). The main advantage of using a circular doubly linked list is that it makes search operation twice as efficient.

Inserting a new node in the Circular Doubly Linked List

In this section, we will see how a new node is added into an already existing circular doubly linked list. We will take two cases and then see how insertion is done in each case. Rest of the cases are similar to that given for doubly linked lists.

- Case 1: The new node is inserted at the end.
- Case 2: The new node is inserted at the beginning.

Consider the circular doubly linked list shown in *Figure 2*. Suppose we want to add a new node with data 9 as the last node of the list. Then the following changes will be done in the linked list.

- Allocate memory for the new node and initialize it's data.
- Take a pointer 'ptr' and make it point to the last node.
- Set 'ptr->next' to point to new node.
- Make the 'next' of the new node point to the head.
- Make the 'prev' of the new node point to ptr.
- Make the 'prev' of the head point to the new node

Read the book for further discussion on inserting items in a Circular Doubly Linked List.

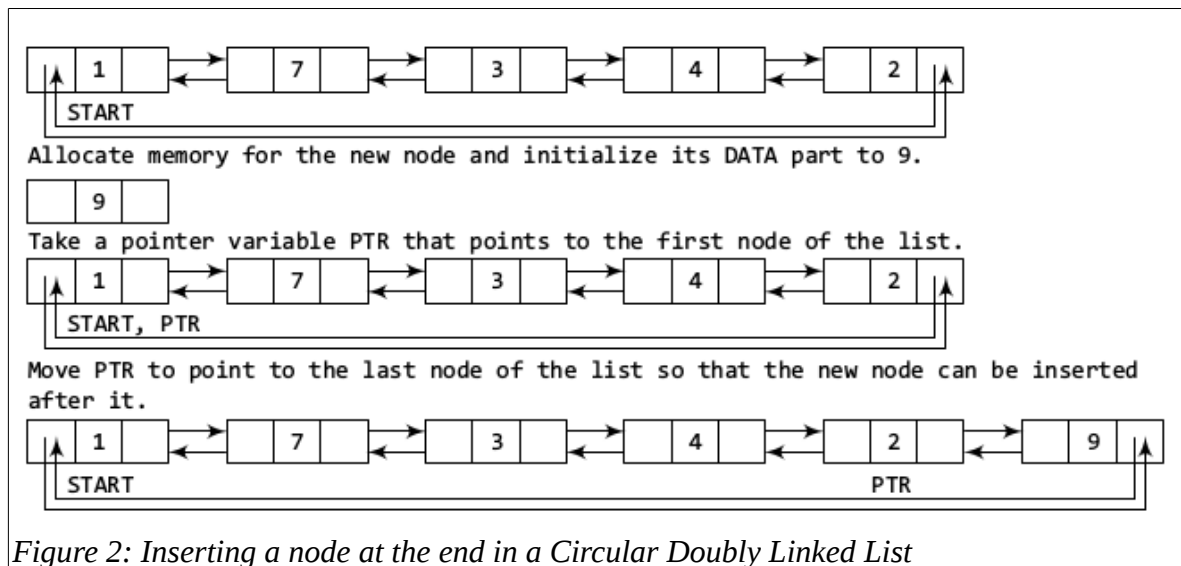


Figure 2: Inserting a node at the end in a Circular Doubly Linked List

Deleting a Node from the End of a Circular Doubly Linked List

Consider the circular doubly linked list shown in *Figure 3*. Suppose we want to delete the last node from the linked list, then the following changes will be done in the linked list.

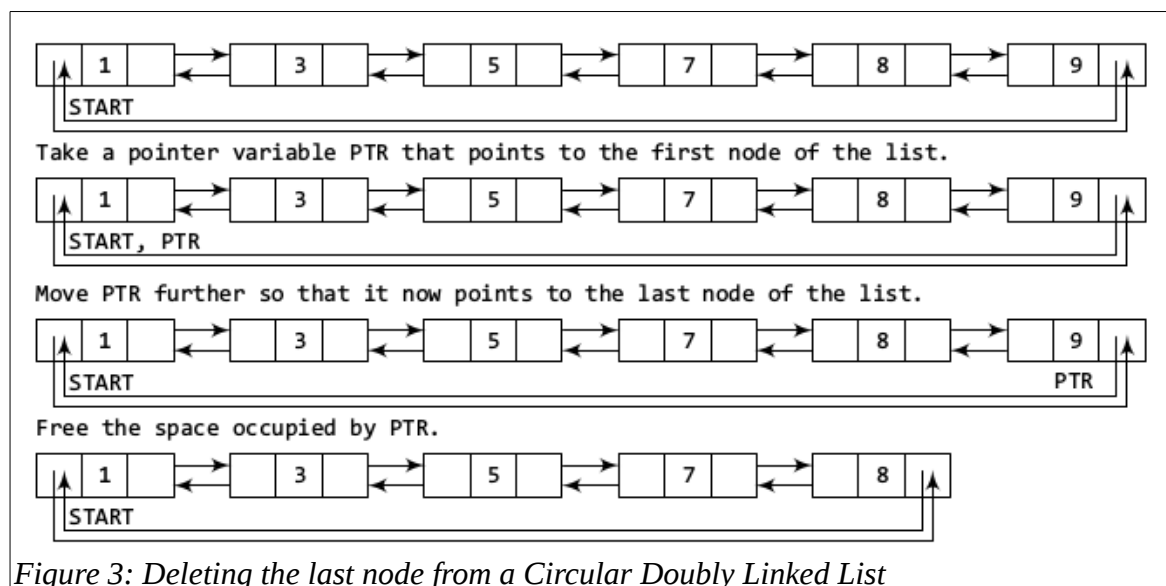


Figure 3: Deleting the last node from a Circular Doubly Linked List

The algorithm to delete the last node from a circular doubly linked list will be as follows.

- Take a pointer variable ptr and initialize it with head. '*ptr*' now points to the first node of the linked list.
- Traverses through the list and make '*ptr*' point to the last node.
- We can also access the second last node from the *prev* field of *ptr*.
- Set the *next* field of the second last node to contain the address of head, so that it now becomes the (new) last node of the linked list.
- Set the *prev* of the head to point to the new last node.
- Free the memory of the last node and return it to the free pool.

Read the book for further discussion on deleting items from a Circular Doubly Linked List.

In Lab Tasks

For this lab you are given the following files:

1. main.c
2. Node2.h
3. DoublyLinkedList.h
4. DoublyLinkedList.c
5. EmployeeData.dat

Task01: Debugging code for errors.

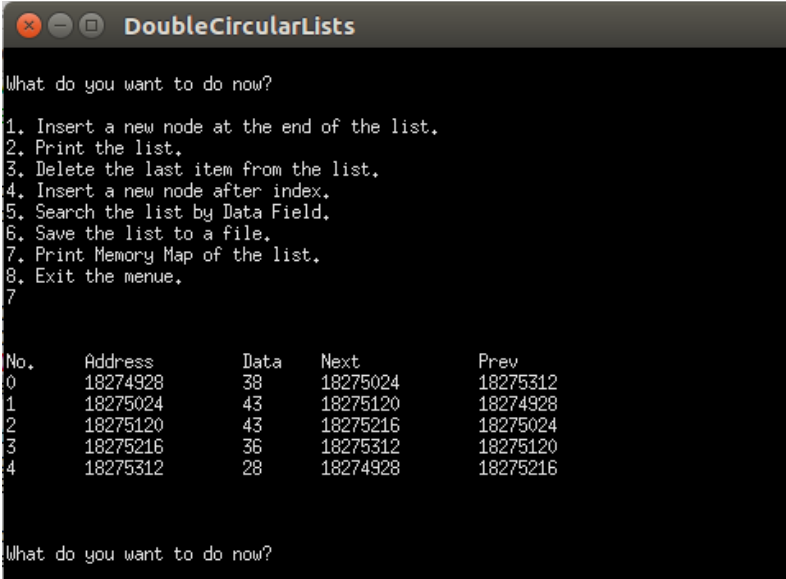
- Make a new project and add these files to the project.
- Compile and run the program
- Choose to load the list from the file

Now you will observe that some of the functions do not work as intended. Figure out the problems (by debugging the code) and correct them so that you have a correctly running implementation of Circular Doubly Linked List. **Report the errors that you have corrected.**

Task 02: Implementing Node Removal and Node Insertion Tasks

Use the created circular linked list in Task 01 and add functions for

1. void deleteNodeFromStart(struct node_d ** head)
2. int deleteNodeAfter(struct node_d * head, int idx)
3. void insertNodeAtStart(struct node_d ** head)



```
What do you want to do now?
1. Insert a new node at the end of the list.
2. Print the list.
3. Delete the last item from the list.
4. Insert a new node after index.
5. Search the list by Data Field.
6. Save the list to a file.
7. Print Memory Map of the list.
8. Exit the menu.
7

No.    Address    Data    Next      Prev
0      18274928    38      18275024  18275312
1      18275024    43      18275120  18274928
2      18275120    43      18275216  18275024
3      18275216    36      18275312  18275120
4      18275312    28      18274928  18275216

What do you want to do now?
```

Figure 4: Printing Memory Map of the linked list

Moreover write a function 'void printMemMap(struct node_d * head)' that prints out the memory map of the linked list as shown in Figure 4.

Post Lab Task: Learn more about at the Josephus Problem from the following links and make a Circular Doubly Linked List simulation of this problem. (Your program should print the remaining people in each iteration).

1. <https://www.youtube.com/watch?v=uCsD3ZGzMgE>
2. <https://github.com/alextop30/Josephus-Problem>

***** End of Document *****