PROGRAM 3
IMPLEMENT CIRCULAR LINKED LIST

ALGORITHM

Step 1: Start
Step 2: declare structure node with int data and node* next.
Step 3: Initialize node* 'front' as NULL.
Step 4: Print Menu
1. Insert at beginning
2. insert at end
3. delete from beginning
4. delete from last
5. exit

Step 5: input user 'choice'.
Step 6: Switch Choice with case =
1. insert_begin().
2. insert_last().
3. delete_begin().
4. delete_last().
5. Exit

Step 6.1 Default Enter a valid choice
Step 7: Stop

insert_begin() :

1. input the element to be inserted and create a node with the entered value.
2. If (front == NULL)
   front = node; node->next =front
3. else
   1. make temp = front
   2. traverse till the end of the list and make temp the last node
   3. make node->next = front.
   4. temp->next = node.
   5. Front = node.
4. Execute display().
5. End.

insert_last() :

1. input the element to be inserted and create a node with the entered value.
2. if(front == NULL)
   front = node;
   node->next = front;
3. else
   1. make temp = front.
   2. Traverse till the end of the list.
   3. Make temp->next = node.
   4. Make node->next = front.
4. Execute display().
5. End.

delete_begin() :

1. check for underflow condition and exit if true.
2. if(front->next = front)
    1. front = NULL
    2. free(front).
    3. Else
    1. make ptr = front.
    2. Traverse till the end of the list
    3. ptr->next = front->next
    4. free(front);
    5. front = ptr->next.
    4. Execute display().
    5. End

delete_last() :

1. check for underflow condition and exit if true.
2. if(front->next == front)
    1. make front = NULL.
    2. free(front)
    3. else
    1. make ptr = front.
    2. Traverse till the end of list and make ‘temp’ the last but one node and ‘ptr’ the last node.
    3. Make temp->next = ptr->next.
    4. free(ptr).
    4. Execute display().
    5. End.