Singly Circular Linked List – This is a special type of Linked List in which the last node of the linked list points to the 1st node of the linked list. Due to this, it is possible to traverse the linked list again and again & also we can consider any node as 1st node of Linked list. This is particularly useful for searching data in a Linked List.

-25 1024

6 600

13 156

18 956

head = 600 956 1024 156

Create a New Singly Circular LL

1st node of LL Subsequent node/s

18 ~~600~~

956

-25 600

18 600

newnode = 600 head = 600 newnode = 956

head temp

temp

temp->next = newnode;

newnode->next = newnode; newnode->next = head;

head = temp = newnode;

Code for Traversal of SCLL

18 956

-25 1024

13 156

6 600

head = 600 956 1024 156

~~temp~~ ~~temp~~ ~~temp~~ temp

temp = head;

// validation is already done that head != NULL & which means LL exists

while (1)

{

printf(“%d “,temp->data); // 18 -25 13 6

temp = temp->next; // 956 1024 156 600

if (temp == head) // this means that after the 1st complete traversal, temp

break; // is again @ position 1 which means we need // to stop

}

**OR**

temp = head;

do

{

printf(“%d “,temp->data);

temp = temp->next;

} while (temp!=head);

13 156

-25 1024

18 956

To Add a Node at the beginning of SCLL

6 ~~600~~  117

head = 600 956 1024 156

temp temp temp temp

37 600

newnode = 117 (This will become the new head of LL)

newnode = (struct node \*)malloc(sizeof(struct node));

printf(“Enter data for 1st node :- “);

scanf(“%d”,&no);

newnode->data = no;

newnode->next = head;

// go to last node & wait

temp = head; // 600

while (temp->next != head) // 956 != 600 T, 1024 != 600 T, 156 != 600 T, 600 != 600 F

temp = temp->next; // 956, 1024, 156

temp->next = newnode;

return newnode; // head = newnode; return head;

18 956

-25 1024

13 156

To Add a Node at the end of LL

6 600 715

head = 600 956 1024 156

temp temp temp temp

13 600

newnode = 715

newnode->next = head;

// come to current last node of LL & change it’s link

temp = head;

while (temp->next != head)

temp = temp->next;

temp->next = newnode;

struct node \*deleteatbeg(struct node \*head)

{

struct node \*temp = head;

// Case – 1, only 1 node in LL

if (head->next == head)

{

free(head);

printf(“Node deleted successfully”);

return NULL;

}

// Case – 2, Multiple nodes in LL

// move temp to last node

while (temp->next != head)

6 156

temp = temp->next;

head = 156

temp->next = head->next; temp

head = head->next;

free(temp);

return head;

}