* LAB-8. Singly Linked List:

→ Insert Front, delet Rear and display
→ Count, and sorting

```
#include<stdio.h>
#include <stdlib.h>

struct node
   { int info;
     struct node *link;
   }
 typedef struct node *NODE;
```

```
struct node {
        int info;
        struct node *link ;
  };


typedef struct node * NODE ;

NODE getnode() {

   NODE x ;
  x = (NODE) malloc (sizeof (struct node));
  if (x == NULL)
    {
        pf ("memory full");
      }
       exit(0);
    }
   return x;


void freenode (NODE x)
      {
        free (x);
      }
```

```
NODE insert_front (NODE first, int

    NODE temp;
    temp = getnode ();
    temp -> info = item;
    temp -> link = NULL;
     if (first == NULL)
            return temp;
      temp -> link = first;
      first = temp;
        return first;
   }
```

```
NODE delete_rear (NODE first) {

    NODE cur, prev ;

    if (first == NULL) {
                PF("EMPTY");
                return first ;
            }

    if (first -> link == NULL) {

                PF(" deleted item is %d, first -> info);
            free (first) ;
            return NULL;

    }

    prev = NULL;
    cur = first;

    while (cur -> link != NULL) {
    prev = cur ;
    cur = cur -> link
    }

    PF(" item deleted is %d", cur -> info);

    fre (cur) ;
    preve -> link = NULL;

    return first;

}
```

```
void display (NODE first) {

    NODE temp;
    if (first ==NULL) {
            pf (" EMPTY LIST ");
        }

    for(temp=first ; temp!=NULL; temp

        {
            pf ("%d , temp->info);
        }

}
```

```
void search (int key, NODE first) {

    NODE   cur;
    if (first == NULL) {
              pf (" list is EMPTY");
            return;
            }


    cur = first;
    while( cur != NULL)
        {
          if (key == cur->info) break;
            cur = cur->link;
          }

     if (cur == NULL)
         {
            pf(" UNSUCCESSFULL");
            return;
         }
     pf ("  SUCCESSFULL ");
}
```

```
[illegible] count (NODE first){

    NODE cur;
    int count = 0;

  if (first==NULL){
              pf(" length zero");
         [illegible] return 0;}

    cur = first

not necessary { if(cur->link == NULL){
                     pf("one element");
               }     return 1;

   while (cur != NULL){

        cur = cur->link;
        count++;
    }

   return count;

}
```

```
order_list (int item, NODE first) {

    NODE  temp, prev, cur;
    temp = getnode ();

    temp -> info = item;
    temp -> link = NULL;

none/
no elements //   if (first == NULL) return temp;

item
less than //     if ( item < first -> info)
first               {
                        temp -> link = first;
                        return temp;
                    }

    prev=NULL;
    cur = first;

rest of the
condition ///
    while (cur != NULL && item > cur -> info)
        {
            prev = cur;
            cur = cur -> link;
        }

    preve -> link = temp;
    temp -> link = cur;

    return first;

}
```

```
int main(){
        int item, choi's, count;
        NODE first =NULL;
        int n, i;
        NODE  a, b;

   for (;;)
       {
       printf("\n 1. Insert-front \n 2. delet
                 rear \n 3. display\n
                  4. count items \n 5.
                    Sort list \n");

 pf("enter choice");
  sf("%d", &choice);
switch (choice) {
    case 1 : printf ("enter the items at
                          front-end \n");
       scf (" %d", &item);
     first = insert-front (first,item);
       break;

   case 2 : first = delet-rear (first);
              break;

   case 4 :  count = length (first);
      pf ("length (items) in the list is %d \n")
                 count);
            break;

   case 5 : printf (" enter the item to be
                      inserted in orderd list);
          sf (" %d", item);
      first = order-list (item, first);
          break;

     default exit (0);
          break;

    }

 }

}
```