Lab - 8,

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 int info;
 struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if(x==NULL)
    {
       printf("memory fully\n");
       exit(0);
    }
    return x;
}
void freenode(NODE x)
{
  free(x);
}
NODE insert_front(NODE first, int item)
{
    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)
        {
            printf("list is empty cannot delete \n");
            return first;
        }
        if (first -> link == NULL)
        {
            printf("item deleted is %d\n", first -> info);
            free(first);
            return NULL;
        }
         prev = NULL;
        cur = first;
        while (cur -> link != NULL)
        {prev = cur;
        cur = cur -> link;}
      printf(" item deleted at rear-end is %d", cur->info);
      free(cur);
      prev -> link = NULL; return first; }
```

```
void swap (NODE a, NODE b)
{ int temp = a-> info;
  a->info = b-> info;
   b->info = temp;

}
void bubblesort( NODE first)

{    int swapped;
     NODE cur;
     NODE prev = NULL;
     if ( first == NULL)
     {  printf("Empty Linked List \n");
       return;
     }
     do
     {   swapped = 0;
         cur = first;
         while (cur -> link != prev)
         {    if (cur -> info > cur -> link -> info)
              {   swap (cur, cur -> link);
                  swapped = 1;
              }
              cur = cur -> link;
         }
          prev = cur;
      } while (swapped);  }
```

```
int list_length(NODE first)
{
    NODE temp;
    int count=0
     if (first == NULL)
      return 0;
     for (temp=first; temp!=NULL; temp=temp->link)
     {
         Count++;
     }
    return count;
}
void search(NODE first, int item)
{
    NODE temp;
    int pos = 0;
    if (first == NULL)
    printf(" list is empty cannot search items \n");
    for (temp = first ; temp != NULL; temp = temp -> link)
    {
        pos ++;
        if (temp -> info == item)
        {
         printf(" Element found and is in the position
                                              %d\n", pos);
        }
    } printf (" Element not found in the list \n");
   return; }
```

Scanned by TapScanner

```
void display(NODE first)
{
    NODE temp;
    if(first == NULL)
      printf("List is empty cannot display items\n");
    for(temp= first; temp != NULL; temp=temp->link)
    {   printf(" %d\n", temp->info);
    }
  }

int main()
{ int item, choice, pos, i, n;
  NODE first = NULL, a,b;
  while(1)
  {
      printf("\n 1. Insert_front \n 2. Delete_rear \n 3. Sort
\n 4. Total items in the list \n 5. Search \n 6. Display
\n 7. Exit \n");
        printf("Enter your choice\n");
        scanf("%d",& choice);
        switch(choice)
        {
          case 1:  printf("Enter the item at the front-end\n");
                   scanf("%d",& item);
                 first= insert-front(first, item);
                   break;
case 2: first = delete-rear(first);
        break;
case 3: bubble_sort(first);
        printf("Items in sorted order are \n");
        display(first);
        break;
Case 4: printf(" Total items in the list is %d \n",
list_length(first)); break;
Case 5: printf(" Enter the item that you want to
search: \n"); scanf("%d", &item); search(~~[illegible]~~ first, item);
case 6: printf(" List: \n"); display(first);
   break;
Case 7: exit (0);
default: printf(" Enter correct instruction!!!\n");
   break; } } return 0; }
```