Date __/__/____

Saathi

linkdist programm

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

Struct node
{
  intdata;
 Struct node *next;
};
 typedef struct node *NODE;

 Void swap (NODE a, NODE b);
 NODE getNode()
{
   NODE x;
   x = malloc (sizeof (structnode));
    if (x = NULL)
   {
      exit(0);
   }
    else
   { return x;
   }
}
   NODE insert_front (int item, NODE
                                 start)
{     NODE temp;
      temp = getNode();
      temp -> data = item;
      temp -> next = start;
```

Date __/__/__

Saathi

```
    Start = temp;
    return Start;
}
NODE insert_end (int item, NODE Start)
{
NODE insert_end (int item, NODE Start)
{
    NODE temp, cur;
    temp = getNode();
    temp -> data = item;
    temp -> next = NULL;
    cur = Start;
    while (cur -> next != NULL)
    {
        cur = cur -> next;
    }
    cur -> next = temp;
    return Start;
}
NODE insert_pos (int pos, int item,
                 NODE Start)
{   NODE temp, prev, cur;
    temp = getNode ();
    temp -> data = item;
    temp -> next = NULL;
    if (pos == 1 && Start == NULL)
    {
        return temp;
    }
    if (Start == NULL)
    {
        Printf ("Invalid postion\n");
        return Start;
    }
    if (pos == 1)
```

Date ___/___/___

```
}
    temp->next = start;
    return temp;
}
    int count = 1;
    cur = start;
    prev = NULL;
    while (cur != NULL && count != pos)
    {
        prev = cur;
        cur = cur->next;
        count++;
    }
    if (count == pos)
    {
        prev->next = temp;
        temp->next = cur;
        return start;
    }
    Printf ("Invalid position\n");
    return start;
}

    NODE temp;
    if (start == NULL)
    {
        printf ("List Is empty And Cannot Be
                 Displayed\n");
        exit (0);
    }
        temp = start;
        Printf("items are \n");
        printf ("\n");
    for (temp = start; temp != NULL;
```

Date __/__/____

```
temp = temp -> next)
{
  Printf ("%d\n", temp -> data);
}
Printf ("\n");
return Start;
}
Void bubbleSort (NODE Start)
{
 int swapped, i;
 NODE cur;
 NODE Prev = NULL)
{
  Printf ("Empty LinKed List\n");
  Return;
}
  do
{
  Swapped = 0;
  cur = Start;
 while (cur->next != Prev)
{
  if (cur -> data > cur -> next -> data)
  }
  Swap (cur, cur -> next);
  Swapped = 1;
  }
  Cur = cur -> next;
}
  Prev = cur;
}
 while (swapped);
}
 Void swap (NODE a, NODE b)
```

```
{
    int temp = a -> data;
    a -> data = b -> data;
    b -> data = temp;
}
Void reverseList (NODE Start)
{
    NODE Prev, Cur;
    if (Start != NULL)
    {
     Prev = Start;
     Start = Start -> next;
     cur = Start;
     Prev => next = NULL; // Make first Node as
                             last node

    while (Start != NULL)
    {
     Start = Start -> next;
     Cur -> next = Prev;
     Prev = Cur;
     Cur = Start;
    }
    Start = Prev; // make last node as head
    Print ("SUCCESSFULLY REVERSED LISTIN");
}
    NODE temp;
    temp = Start;
    printf("Reversed List is \n");
    while (temp != NULL)
    {
     Printf ("%d\n", temp -> data);
     temp = temp -> next;
    }
}
```

Date ___/___/___

```
int main()
{
 int option, item, pos;
 NODE start = NULL;
 do
 {
 Printf("1: Insert Front\n");
 Printf("2: Insert End\n");
 Printf("3: Insert At particular position\n");
 Printf("4: Delete First\n");
 printf("5: Delete End\n");
 Printf("6: Delete At particular position\n");
 Printf("7: Display\n");
 Printf("8: Sort\n");
 Printf("9: Reverse\n");
 Printf("10: exit\n");
 Printf("Enter your choice\n");
 Scanf (option) ("%d", &option);
 Switch (option)
 {
  case 1:
   printf("Enter the item to be inserted\n");
   Scanf("%d", &item);
   start = insert_front(item, start);
   break;
  case 2:
        printf("Enter the item to be
                inserted\n");
       Scanf("%d", &item);
       Start = insert_end(item, start);
        break;
    case 3:
    Printf("Enter the item to be inserted
```

Date ___/___/___

```
scanf ("%d", &pos);
start = insert_pos (pos, item, start);
break;

case 4;
    start = delete_front (start);
    break;

case 5:
    start_delete_end (start);
    break;

case 6:
    printf ("Enter the position where element
             has to be inserted \n");
    scanf ("%d", &pos);
    start = delete_pos (pos, start);
    break;

case 7:
    start = display (start);
    break;

case 8:
    bubblesoart (start);
printf (" Items In Sorted order are\n
 start = display (start);
 break;

Case 9;
    reverselist (start);
    break;
}
} while (option != 10);
}
```

Date ___/___/_____

Saathi

```
Node reverse(Node first)
{
Node cur, prev, temp;
if(first==NULL)
{
printf("List is Empty\n");
return first;
}
cur = NULL;
while(first!=NULL)
{
temp = first;
first = first->next);
temp-> next = cur;
cur = temp;
}
return cur;
}
void sort(Node first)
{
int t;
Node temp;
if(first==NULL)
{
printf("List IS Empty\n");
return;
}
for(Node i=first; i!=NULL; i=i->next)
{
for(Node j=i->next; j!=NULL
                j=j->next)
{
if((i->item) > (j->item))
```

Saathi

Date __/__

```
        t = i -> item;
        i -> item = j -> item
        j -> item = t;
    }
}

Node concat (Node first, Node second)
{
    Node cur;
    if (first == NULL)
    {
        first = second;
        return first;
    }
    if (second == NULL)
    {
        return first;
    }
    cur = first;
    while (cur -> next != NULL)
    {
        cur = cur -> next;
    }
    cur -> next = second;
    printf("Final Concatinated List is\n");
    return first;
}
```