25/11/2020

PROGRAM-1 & 2

```
#include <stdio.h>
 struct node
{
 int data;
 struct node * next;
};
 struct node * head = NULL;
 int length=0;
 void insertend (int ele)
 {
  struct node* newnode, *temp;
  newnode = (struct node*) malloc sizeof(struct node));
   temp= head;
   while (temp->next != NULL)
   {
    temp= temp->next;
   }
   temp->next = newnode;
   length++;
  }
  void insertionfront (int ele)
  {
   struct node * temp;
    temp = (struct node *) malloc (sizeof (struct node));
    temp->data = ele;
    temp->next = head;
    head = temp;
     length++;
   }
```

```
void insertrandom (int ele, int pos)
{
  if (pos == 1)
   insertfront (ele);
   else  if (pos >= length)
    insert end (ele);

    else
   {
    struct node *inst;
    inst = (struct node *) malloc (sizeof (struct node));
     struct node * temp;
      temp = (struct node *) malloc (sizeof (struct node));
       temp = head;
       for (int i = 1; i < pos-1 ; i++)
        {
          temp = temp->next;
         }
          inst ->data = ele;
         inst -> next = temp->next;
          temp -> next = inst;
           length++;
        }

      }
      void deleteele (int ele)
     {
      struct node * temp, * del;
      temp = (struct node*) malloc (sizeof (struct node));
       del = (struct node*) malloc (sizeof (struct node));
        del = NULL;
        if (head->data == ele)
         {
```

```
    del= temp->next;
    temp->next=del->next;
    del->next = NULL;
     length--;
     break;

  }
  else
  {
  temp= temp->next;

  }
 }
}
if (del==NULL)
{
 printf("\nElement not found.\n");
 }
}

void display()
  {
   struct node* temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp= head;
    if( temp==NULL)
     {
      printf("\nList is empty\n");
      }
     else
     {
      printf("\nThe contents of the list are:\n");
      while (temp!=NULL)
```

```
{
    printf("%d\n", temp->data);
    temp= temp->next;
  }
 }
}
int main()
{
int choice, ele,pos;
char ch;
do
{
 printf("\n 1.Insert at end \n2.Insert at front \n3. Insert
  at random position \n4. Display \n5. Delete \n6. Exit ");
  printf("\nEnter your choice : ");
  scanf("%d", &choice);
   switch (choice)
  {
    case 1: printf(" Enter the element to be inserted \n");
           scanf("%d ", &ele);
           insertend (ele);
           break;
     case 2: printf(" Enter the element tobe inserted \n");
            scanf("%d", &ele);
            insertatfront (ele);
             break;
     case3: printf(" Enter the element to be inserted\n");
            scanf("%d ",&ele);
            printf(" Enter the position \n");
```

```
void display (NODE first)
{
 NODE temp;
 if (first==NULL)
 printf ("list empty");
 for (temp=first; temp!=NULL; temp=temp->link)
 {
 printf ("%d\n", temp->info);
 }
 }
 NODE concat (NODE first, NODE second)
 {
  NODE cur;
  if( first==NULL)
  return second;
  if(second==NULL)
  return first;
   cur=first;
   while (cur->link!=NULL)
   cur=cur->link;
   cur->link=second;
    return first;
   }
    NODE reverse (NODE first)
   {
   NODE cur, temp;
   cur=NULL;
   while (first!=NULL)
  {
  temp=first;
  first=first->link;
```

Scanned with CamScanner

```
temp->link = cur;
 cur = temp;
}
 return cur;
}
int main()
{
 int item, choice, pos, i, n;
 NODE first = NULL, a, b;
 for(;;)
 {
 printf("1. insert_front\n2. concat\n3. reverse\n4. display\n5. exit\n");
 printf("enter the choice\n");
 scanf("%d", &choice);
  switch (choice)
  {
   case 1: printf("enter the item\n");
    scanf("%d", &item);
     first = insert-rear(first, item);
     break;

   case 2: printf("enter the no of nodes in 1\n");
    scanf("%d", &n);
     a = NULL;
     for(i=0; i<n; i++)
     {
     printf("enter the item\n");
     scanf("%d", &item);
      a = insert-rear(a, item);
     }
printf("enter the no of nodes in 2\n");
scanf("%d", &n);
 b= NULL;
 for (i=0; i<n; i++)

{
printf("enter the item\n");
scanf("%d", &item);
 b= insert-rear(b, item);

}
 a= concat(a, b);
 display(a);
  break;
  case 3: first = reverse (first);
  display (first);
   break;
   case 4: display (first);
   break;
    default: exit(0);
   }
  }
 }
```