```
# include <stdio.h>
# include <stdlib.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 (size of (struct node));
     newnode -> data = ele;
     newnode -> next = NULL;
      if (head == NULL)
    {
        head = newnode;
        length = 1;
     }
     else
    { temp = temp -> next;
     }
        temp -> next = newnode;
        length ++;
     }

   }
```

```
void insertfront (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+1 ) ;
   ~~el~~ .  insertend (ele ) ;
   else
   {
       struct node * inst ;
     inst ( struct node *) malloc (size of (struct node));
     struct node * temp ;
   temp = (struct node *) malloc (size of (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 deletefront()
{
    if (length==0)
    { printf("\n List is empty. \n");

    }
    else
    { struct node *temp;

        temp = (struct node*) malloc (size of (struct node));
        temp = head;
        head = head ->next;
        temp -> next = NULL;
        length--;
        printf("\n The element deleted is %d", temp->data)
    }
}

void deleteend()
{
    if (length==0)
    { printf("\n List is empty. \n");
    }
    else
    { struct node *temp;

        temp = (struct node*)malloc (size of (struct node));
        temp= head;
```

```
while ( temp → next → next != NULL)
 {
     temp = temp → next ;
  }
 struct node * del ;
 del = (struct node*) ~~malloc (sizeof~~
 malloc (sizeof (struct node)) ;
 del = temp → next ;
 temp → next = NULL;
 length --;
 printf ("\n The element deleted is : %d", del → data);
}
}

Void  deleterandom (int pos)
  if (length = =0)
  printf (" List is empty \n");
 else if (pos = = 1)
     deletefront () ;
 else if (pos >= length + 1)
   deleteend () ;
 else
 {
    struct node * del ;
  del = (struct node * ) malloc( size of (struct node));
```

```
struct node * temp;
temp = (struct node *) malloc (size of ( struct node));
temp = head ;
 for (int i=1 ; i<pos-1 ; i++ )
 {
        temp = temp-> next ;

  }
    del = temp -> next ;
    temp -> next = del -> next ;
   del -> next = NULL ;
   length -- ;
 printf (" \n The element deleted is : %d , del -> data);
  }
 }
 void display ( )
 { struct node * temp ;
   temp = (struct node *) malloc ( size of ( struct node));
  temp = head ;
   if ( temp = = NULL )
  {
   printf (" \n List is empty \n ) ;
   }
   else
```

```
{
 printf ("\n The 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 \n 2. Insert at front
     \n 3. Insert at random position \n 4. Display
   \n 5. Delete at front \n 6. Delete at end \n 7.
   Delete at random \n 8. Exit");
  printf ("\n Enter 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 to be inserted \n");
        scanf ("%d", &ele);
        insertfront (ele);
        break;
Case 3:  printf ("Enter the element to be inserted \n");
         scanf ("%d", &ele);
        printf (" Enter the position \n");
         scanf ("%d", &pos);
        insertrandom (ele, pos);
        break;
 Case 4;  display ()
            break;
 case 5;  deletefront ()
          break;
 case 6:   deleteend ();
            break;
 Case 7:  printf ("\nEnter the position:");
          scanf ("%d", & pos);
         deleterandom (pos);
          break;
      }
    } while (choice != 0)
      return 0;
  }
```