Assessment-2: Linked List

Reg.no: 20BDS0117

Name: SANJANA.SAIRAMA

Slot: L37+L38

Aim/Objective: Write C program for the questions using linked lists.

1) Write a program in C to perform the following operation in a singly linked list.

i) Insert a new node at the beginning of a Singly Linked List.

ii) Insert a new node at the end of a Singly Linked List.

iii) Insert in the middle of the linked list.

Pseudo code :

1.Begin

2.Initialize the structure and variables

```
3. procedure createList(int n)
   struct node *newNode, *temp
   int data, i
   head = (struct node *)malloc(sizeof(struct node))
   print Enter the data
   scanf("%d", &data)
   head->data = data
   head->next = NULL
   temp = head
   for (i = 2; i <= n; i++)
      newNode = (struct node *)malloc(sizeof(struct node))
      print Enter the data of node
      scanf("%d", &data)
      newNode->data = data;
      newNode->next = NULL;
      temp->next = newNode;
      temp = temp->next;
   endfor
  Display the linked list
  End procedure
```

```
4. procedure insertNodeAtBeginning(int data)
     struct node *newNode
     newNode = (struct node *)malloc(sizeof(struct node))
     newNode->data = data
     newNode->next = head
```

```
    head = newNode
    print DATA INSERTED SUCCESSFULLY
end procedure
```

5.procedure insertNodeAtEnd(int data)

```
  struct node *newNode
  struct node *ptr
  ptr = head
  newNode = (struct node *)malloc(sizeof(struct node))
  while (ptr != NULL && ptr->next != NULL)
     ptr = ptr->next
  endwhile
  newNode->data = data
  newNode->next = NULL
  ptr->next = newNode
  print DATA INSERTED SUCCESSFULLY
end procedure
```

6.Procedure insertNodeAtMiddle(int data, int position)

```
  int i
  struct node *newNode, *temp
  newNode = (struct node *)malloc(sizeof(struct node))
  newNode->data = data
  newNode->next = NULL
  temp = head
  for (i = 2; i <= position - 1; i++)
     temp = temp->next
     if (temp == NULL)
        break
     endif
  endfor
  if (temp != NULL)
     newNode->next = temp->next
     temp->next = newNode
     print DATA INSERTED SUCCESSFULLY
  endif
  else
     print UNABLE TO INSERT DATA AT THE GIVEN POSITION
  endelse
endprocedure
```

7. procedure displayList()

```
  struct node *temp
  if (head == NULL)
     Print List is empty.
  endif
  else
```

```
      temp = head
      while (temp != NULL)
         print("Data = %d\n", temp->data)
         temp = temp->next
      endwhile
   endelse
end procedure
```

8.Input the values for the required procudures

9.End

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
} * head;
void createList(int n);
void insertNodeAtBeginning(int data);
void displayList();
void insertNodeAtEnd(int data);
void insertNodeAtMiddle(int data, int position);

void createList(int n)
{
   struct node *newNode, *temp;
   int data, i;
   head = (struct node *)malloc(sizeof(struct node));

   printf("Enter the data of node 1: ");
   scanf("%d", &data);

   head->data = data;
   head->next = NULL;

   temp = head;
   for (i = 2; i <= n; i++)
   {
      newNode = (struct node *)malloc(sizeof(struct node));

      printf("Enter the data of node %d: ", i);
      scanf("%d", &data);

      newNode->data = data;
```

```
        newNode->next = NULL;
        temp->next = newNode;
        temp = temp->next;
    }
    printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
}

void insertNodeAtBeginning(int data)
{
    struct node *newNode;
    newNode = (struct node *)malloc(sizeof(struct node));

    newNode->data = data;
    newNode->next = head;
    head = newNode;

    printf("DATA INSERTED SUCCESSFULLY\n");
}

void displayList()
{
    struct node *temp;
    if (head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while (temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
}
void insertNodeAtEnd(int data)
{
    struct node *newNode;
    struct node *ptr;
    ptr = head;
    newNode = (struct node *)malloc(sizeof(struct node));
    while (ptr != NULL && ptr->next != NULL)
    {
        ptr = ptr->next;
```

```
    }
    newNode->data = data;
    newNode->next = NULL;
    ptr->next = newNode;
    printf("DATA INSERTED SUCCESSFULLY\n");
}

void insertNodeAtMiddle(int data, int position)
{
    int i;
    struct node *newNode, *temp;
    newNode = (struct node *)malloc(sizeof(struct node));

    newNode->data = data;
    newNode->next = NULL;

    temp = head;
    for (i = 2; i <= position - 1; i++)
    {
        temp = temp->next;
        if (temp == NULL)
            break;
    }

    if (temp != NULL)
    {
        newNode->next = temp->next;
        temp->next = newNode;

        printf("DATA INSERTED SUCCESSFULLY\n");
    }
    else
    {
        printf("UNABLE TO INSERT DATA AT THE GIVEN POSITION\n");
    }
}

int main()
{
    int n, data,pos;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
```

```
    displayList();

    printf("\nEnter data to insert at beginning of the list: ");
    scanf("%d", &data);
    insertNodeAtBeginning(data);

    printf("\nData in the list \n");
    displayList();

    printf("\nEnter data to insert at end of the list: ");
    scanf("%d", &data);
    insertNodeAtEnd(data);

    printf("\nData in the list \n");
    displayList();

    printf("\nEnter data to insert at middle of the list: ");
    scanf("%d", &data);
    printf("\nEnter position: ");
    scanf("%d", &pos);
    insertNodeAtMiddle(data,pos);

    printf("\nData in the list \n");
    displayList();

    return 0;
}
```

Output

"C:\Users\USER\Downloads\linked list\bin\Debug\linked list.exe"

```
Enter the total number of nodes: 5
Enter the data of node 1: 24
Enter the data of node 2: 65
Enter the data of node 3: 88
Enter the data of node 4: 76
Enter the data of node 5: 98
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 24
Data = 65
Data = 88
Data = 76
Data = 98

Enter data to insert at beginning of the list: 12
DATA INSERTED SUCCESSFULLY

Data in the list
Data = 12
Data = 24
Data = 65
Data = 88
Data = 76
Data = 98

Enter data to insert at end of the list: 34
DATA INSERTED SUCCESSFULLY

Data in the list
Data = 12
Data = 24
Data = 65
Data = 88
Data = 76
Data = 98
Data = 34
```

```
Enter data to insert at middle of the list: 45

Enter position: 3
DATA INSERTED SUCCESSFULLY

Data in the list
Data = 12
Data = 24
Data = 45
Data = 65
Data = 88
Data = 76
Data = 98
Data = 34

Process returned 0 (0x0)   execution time : 59.831 s
Press any key to continue.
```

2) Write a program in C to perform the following operation in a singly linked list.

i) To delete a node at the beginning of the list

ii) To delete a node in the middle of the linked list

iii) To delete at the end of the linked list

Pseudo code (i)

```
1.Begin
2.Initialize the structure and the variables
3. procedure  createNodeList(int n)
   struct node *fnNode, *tmp
   int num, i
   stnode = (struct node *)malloc(sizeof(struct node))
   if(stnode == NULL)
      print Memory can not be allocated.
   endif
   else
      Print Input data for node 1
      scanf("%d", &num)
      stnode-> num = num
      stnode-> nextptr = NULL
      tmp = stnode
      for(i=2; i<=n; i++)
         fnNode = (struct node *)malloc(sizeof(struct node))
         if(fnNode == NULL)
            Print Memory can not be allocated.
            break
         endif
         else
            print Input data for node %d : ", i
            scanf(" %d", &num)
            fnNode->num = num
```

```
            fnNode->nextptr = NULL
             tmp->nextptr = fnNode
            tmp = tmp->nextptr
          endelse
      endfor
   endelse
endprocedure
```

4. procedure deleteatbeg()

```
   struct node *toDelptr
   if(stnode == NULL)
      print There are no node in the list.
   endif
   else
      toDelptr = stnode
      stnode = stnode->nextptr
      printf("\n Data of node 1 which is being deleted is :  %d\n", toDelptr->num)
      free(toDelptr)
   endelse
end procedure
```

5.procedure displayList()

```
   struct node *tmp
   if(stnode == NULL)
      Print No data found in the list.
   endif
   else
      tmp = stnode
      while(tmp != NULL)
         printf(" Data = %d\n", tmp->num)
         tmp = tmp->nextptr
      endwhile
```

```
   endelse
end procedure
6.End
```

Code (i)

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

struct node
{
   int num;
   struct node *nextptr;
}*stnode;

void createNodeList(int n);
void deleteatbeg();
void displayList();

int main()
{
   int n,num,pos;
   printf(" Input the number of nodes : ");
   scanf("%d", &n);
   createNodeList(n);
   printf("\n Data entered in the list are : \n");
   displayList();
   deleteatbeg();
   printf("\n Data after deletion of first node : \n");
   displayList();
   return 0;
}
```

```
void createNodeList(int n)
{
   struct node *fnNode, *tmp;
   int num, i;
   stnode = (struct node *)malloc(sizeof(struct node));
   if(stnode == NULL)
   {
      printf(" Memory can not be allocated.");
   }
   else
   {
      printf(" Input data for node 1 : ");
      scanf("%d", &num);
      stnode-> num = num;
      stnode-> nextptr = NULL;
      tmp = stnode;
      for(i=2; i<=n; i++)
      {
         fnNode = (struct node *)malloc(sizeof(struct node));
         if(fnNode == NULL)
         {
            printf(" Memory can not be allocated.");
            break;
         }
         else
         {
            printf(" Input data for node %d : ", i);
            scanf(" %d", &num);
            fnNode->num = num;
            fnNode->nextptr = NULL;
```

```
            tmp->nextptr = fnNode;
            tmp = tmp->nextptr;
        }
    }
  }
}
void deleteatbeg()
{
  struct node *toDelptr;
  if(stnode == NULL)
  {
    printf(" There are no node in the list.");
  }
  else
  {
    toDelptr = stnode;
    stnode = stnode->nextptr;
    printf("\n Data of node 1 which is being deleted is :  %d\n", toDelptr->num);
    free(toDelptr);
  }
}
void displayList()
{
  struct node *tmp;
  if(stnode == NULL)
  {
    printf(" No data found in the list.");
  }
  else
  {
```

```
    tmp = stnode;
    while(tmp != NULL)
    {
        printf(" Data = %d\n", tmp->num);
        tmp = tmp->nextptr;
    }
  }
}
```

Output:

"C:\Users\USER\Downloads\insert node using ll\bin\Debug\insert node using ll.exe"

```
 Input the number of nodes : 5
 Input data for node 1 : 6
 Input data for node 2 : 7
 Input data for node 3 : 8
 Input data for node 4 : 9
 Input data for node 5 : 2

 Data entered in the list are :
 Data = 6
 Data = 7
 Data = 8
 Data = 9
 Data = 2

 Data of node 1 which is being deleted is :  6

 Data after deletion of first node :
 Data = 7
 Data = 8
 Data = 9
 Data = 2

Process returned 0 (0x0)   execution time : 7.918 s
Press any key to continue.
```

Pseudo code (ii)

1.Begin

2.Initialize the structure and variables

3. procedure main()

int n,num,pos

print Input the number of nodes

scanf("%d", &n)

```
   createNodeList(n)
   print Data entered in the list are
   displayList()
   print  Input the position of node to delete
   scanf("%d", &pos)
   if(pos<=1 || pos>=n)
       print  Deletion can not be possible from that position.
   endif
     if(pos>1 && pos<n)
         print Deletion completed successfully
         MiddleNodeDeletion(pos)
     endif
              print  The new list are
   displayList()
end procedure
4. procedure createNodeList(int n)
   struct node *fnNode, *tmp
   int num, i
   stnode = (struct node *)malloc(sizeof(struct node))
   if(stnode == NULL)
      print Memory can not be allocated.
   endif
   else
      print Input data for node 1
      scanf("%d", &num)
      stnode-> num = num
      stnode-> nextptr = NULL
      tmp = stnode
      for(i=2; i<=n; i++)
         fnNode = (struct node *)malloc(sizeof(struct node))
```

```
        if(fnNode == NULL)
            print  Memory can not be allocated.
            break
        endif
        else
            print Input data for node
            scanf(" %d", &num)
            fnNode->num = num
            fnNode->nextptr = NULL
            tmp->nextptr = fnNode
            tmp = tmp->nextptr
        endelse
    endfor
  endelse
end procedure
```

5. procedure MiddleNodeDeletion(int pos)

```
  int i
  struct node *toDelMid, *preNode
  if(stnode == NULL)
    print There are no nodes in the List.
  endif
  else
    toDelMid = stnode
    preNode = stnode
    for(i=2; i<=pos; i++)
        preNode = toDelMid
        toDelMid = toDelMid->nextptr
        if(toDelMid == NULL)
            break
        endif
```

```
        endfor
        if(toDelMid != NULL)
            if(toDelMid == stnode)
                stnode = stnode->nextptr
            preNode->nextptr = toDelMid->nextptr
            toDelMid->nextptr = NULL
            free(toDelMid)
        endif
        else
            print Deletion can not be possible from that position.
        endelse
    endelse
end procedure
```

6.procedure displayList()

```
    struct node *tmp
    if(stnode == NULL)
        print No data found in the list.
    endif
    else
        tmp = stnode
        while(tmp != NULL)
            printf(" Data = %d\n", tmp->num)
            tmp = tmp->nextptr
        endwhile
    endelse
end procedure
```

7.End

Code (ii)

```
#include <stdio.h>

#include <stdlib.h>


struct node

{

   int num;

   struct node *nextptr;

}*stnode;


void createNodeList(int n);

void MiddleNodeDeletion(int pos);

void displayList();


int main()

{

   int n,num,pos;

   printf(" Input the number of nodes : ");

   scanf("%d", &n);

   createNodeList(n);


   printf("\n Data entered in the list are : \n");

   displayList();


   printf("\n Input the position of node to delete : ");

   scanf("%d", &pos);


   if(pos<=1 || pos>=n)

   {

    printf("\n Deletion can not be possible from that position.\n ");
```

```
    }
                if(pos>1 && pos<n)
     {
      printf("\n Deletion completed successfully.\n ");
             MiddleNodeDeletion(pos);
     }


              printf("\n The new list are  : \n");
    displayList();
    return 0;
}
void createNodeList(int n)
{
    struct node *fnNode, *tmp;
    int num, i;

    stnode = (struct node *)malloc(sizeof(struct node));
    if(stnode == NULL)
    {
        printf(" Memory can not be allocated.");
    }
    else
    {
        printf(" Input data for node 1 : ");
        scanf("%d", &num);

        stnode-> num = num;
        stnode-> nextptr = NULL;
        tmp = stnode;
        for(i=2; i<=n; i++)
```

```
        {
            fnNode = (struct node *)malloc(sizeof(struct node));



            if(fnNode == NULL)
            {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);

                fnNode->num = num;
                fnNode->nextptr = NULL;

                tmp->nextptr = fnNode;
                tmp = tmp->nextptr;
            }
        }
    }
}

void MiddleNodeDeletion(int pos)
{
    int i;
    struct node *toDelMid, *preNode;

    if(stnode == NULL)
```

```
{
    printf(" There are no nodes in the List.");
}
else
{
    toDelMid = stnode;
    preNode = stnode;

    for(i=2; i<=pos; i++)
    {
        preNode = toDelMid;
        toDelMid = toDelMid->nextptr;

        if(toDelMid == NULL)
            break;
    }
    if(toDelMid != NULL)
    {
        if(toDelMid == stnode)
            stnode = stnode->nextptr;

        preNode->nextptr = toDelMid->nextptr;
        toDelMid->nextptr = NULL;
        free(toDelMid);
    }
    else
    {
        printf(" Deletion can not be possible from that position.");
    }
}
```

```
}

void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
        printf(" No data found in the list.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);
            tmp = tmp->nextptr;
        }
    }
}
```

Output

"C:\Users\USER\Downloads\insert node using ll\bin\Debug\insert node using ll.exe"

```
 Input the number of nodes : 5
 Input data for node 1 : 6
 Input data for node 2 : 7
 Input data for node 3 : 8
 Input data for node 4 : 9
 Input data for node 5 : 2

 Data entered in the list are :
 Data = 6
 Data = 7
 Data = 8
 Data = 9
 Data = 2

 Input the position of node to delete : 2

 Deletion completed successfully.

 The new list are  :
 Data = 6
 Data = 8
 Data = 9
 Data = 2

Process returned 0 (0x0)   execution time : 19.514 s
Press any key to continue.
```

Pseudo code (iii)

1.Begin

2.Initialize structure and variables

3. procedure createNodeList(int n)

```
   struct node *fnNode, *tmp

   int num, i

   stnode = (struct node *)malloc(sizeof(struct node))

   if(stnode == NULL)

      Print Memory can not be allocated.

   endif

   else

      print Input data for node

      scanf("%d", &num)

      stnode-> num = num

      stnode-> nextptr = NULL
```

```
        tmp = stnode
        for(i=2; i<=n; i++)
            fnNode = (struct node *)malloc(sizeof(struct node))
            if(fnNode == NULL)
                Print Memory can not be allocated.
                break
            endif
            else
                print Input data for node
                scanf(" %d", &num)
                fnNode->num = num
                fnNode->nextptr = NULL
                tmp->nextptr = fnNode
                tmp = tmp->nextptr
            endelse
        endfor
    endelse
end procedure
4. procedure deleteatend()
    struct node *toDelLast, *preNode
    if(stnode == NULL)
        printThere is no element in the list.
    endif
    else
        toDelLast = stnode
        preNode = stnode
        while(toDelLast->nextptr != NULL)
            preNode = toDelLast
            toDelLast = toDelLast->nextptr
        if(toDelLast == stnode)
```

```
        stnode = NULL
      endif
      else
        preNode->nextptr = NULL
      endelse
      free(toDelLast)
    endelse
end procedure
5.procedure displayList()
    struct node *tmp
    if(stnode == NULL)
      print No data found in the empty list.
    endif
    else
      tmp = stnode
      while(tmp != NULL)
        print(" Data = %d\n", tmp->num)
        tmp = tmp->nextptr
      endwhile
    endelse
end procedure
6.End
```

Code (iii)

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

struct node
{
    int num;
    struct node *nextptr;
```

```
}*stnode;

void createNodeList(int n);
void deleteatend();
void displayList();

int main()
{
   int n,num,pos;
   printf(" Input the number of nodes : ");
   scanf("%d", &n);
   createNodeList(n);

   printf("\n Data entered in the list are : \n");
   displayList();
   deleteatend();
            printf("\n The new list after deletion the last node are  : \n");
   displayList();
   return 0;
}
void createNodeList(int n)
{
   struct node *fnNode, *tmp;
   int num, i;

   stnode = (struct node *)malloc(sizeof(struct node));
   if(stnode == NULL)
   {
      printf(" Memory can not be allocated.");
   }
```

```
    else
    {
        printf(" Input data for node 1 : ");
        scanf("%d", &num);

        stnode-> num = num;
        stnode-> nextptr = NULL;
        tmp = stnode;
        for(i=2; i<=n; i++)
        {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode == NULL)
            {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);
                fnNode->num = num;
                fnNode->nextptr = NULL;
                tmp->nextptr = fnNode;
                tmp = tmp->nextptr;
            }
        }
    }
}
void deleteatend()
{
```

```
    struct node *toDelLast, *preNode;
    if(stnode == NULL)
    {
        printf(" There is no element in the list.");
    }
    else
    {
        toDelLast = stnode;
        preNode = stnode;
        while(toDelLast->nextptr != NULL)
        {
            preNode = toDelLast;
            toDelLast = toDelLast->nextptr;
        }
        if(toDelLast == stnode)
        {
            stnode = NULL;
        }
        else
        {
            preNode->nextptr = NULL;
        }
        free(toDelLast);
    }
}
void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
```

```
        printf(" No data found in the empty list.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);
            tmp = tmp->nextptr;
        }
    }
}
```

Output:

"C:\Users\USER\Downloads\insert node using ll\bin\Debug\insert node using ll.exe"

```
 Input the number of nodes : 5
 Input data for node 1 : 6
 Input data for node 2 : 7
 Input data for node 3 : 8
 Input data for node 4 : 9
 Input data for node 5 : 2

 Data entered in the list are :
 Data = 6
 Data = 7
 Data = 8
 Data = 9
 Data = 2

 The new list after deletion the last node are  :
 Data = 6
 Data = 7
 Data = 8
 Data = 9

Process returned 0 (0x0)   execution time : 10.208 s
Press any key to continue.
```

3) Write a program in C to search an element in the singly linked list.

Pseudo code:

```
1.Begin
2.Initialize the structure and variables
3. void main()
        int n,i,FindElem,FindPlc
        stnode.nextptr=NULL
        ennode=&stnode
  print  Enter the number of nodes
  scanf("%d", &n)
        for(i=0;i< n;i++)
                ennode->nextptr=(struct node *)malloc(sizeof(struct node))
                print(" Enter the data for node %d : ",i+1)
                scanf("%d",&ennode->num)
                ennode=ennode->nextptr
        endfor
        ennode->nextptr=NULL
        print  Data entered in the list are
         ennode=&stnode
        while(ennode->nextptr!=NULL)
                printf(" Data = %d\n",ennode->num)
                ennode=ennode->nextptr
        endwhile
        print  Enter the element to be searched
        scanf("%d",&FindElem)
        FindPlc=FindElement(FindElem)
        if(FindPlc<=n)
                print Element found at node %d \n\n",FindPlc
        else
                print This element does not exists in linked list.
```

```
End procedure
4. procedure FindElement(int FindElem)
        int ctr=1
        ennode=&stnode
        while(ennode->nextptr!=NULL)
                if(ennode->num==FindElem)
                        break
                else
                        ctr increment
                        ennode=ennode->nextptr
        endwhile
        return ctr
end procedure
5.end
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int num;
  struct node *nextptr;
}

stnode, *ennode;
int FindElement(int);
void main()
{
        int n,i,FindElem,FindPlc;
        stnode.nextptr=NULL;
        ennode=&stnode;
```

```
   printf(" Enter the number of nodes : ");
   scanf("%d", &n);
        printf("\n");
        for(i=0;i< n;i++)
        {
                ennode->nextptr=(struct node *)malloc(sizeof(struct node));
                printf(" Enter the data for node %d : ",i+1);
                scanf("%d",&ennode->num);
                ennode=ennode->nextptr;
        }
        ennode->nextptr=NULL;
        printf("\n Data entered in the list are :\n");

   ennode=&stnode;
        while(ennode->nextptr!=NULL)
        {
                printf(" Data = %d\n",ennode->num);
                ennode=ennode->nextptr;
        }

        printf("\n");
        printf(" Enter the element to be searched : ");
        scanf("%d",&FindElem);
        FindPlc=FindElement(FindElem);
        if(FindPlc<=n)
                printf(" Element found at node %d \n\n",FindPlc);
        else
                printf(" This element does not exists in linked list.\n\n");
}
int FindElement(int FindElem)
```

```
{
        int ctr=1;
        ennode=&stnode;
        while(ennode->nextptr!=NULL)
        {
                if(ennode->num==FindElem)
                        break;
                else
                        ctr++;
                        ennode=ennode->nextptr;
        }
        return ctr;
}
```

Output

"C:\Users\USER\Downloads\search an elements using sll\bin\Debug\search an elements using sll.exe"

```
Enter the number of nodes : 5

Enter the data for node 1 : 3
Enter the data for node 2 : 7
Enter the data for node 3 : 6
Enter the data for node 4 : 8
Enter the data for node 5 : 9

Data entered in the list are :
Data = 3
Data = 7
Data = 6
Data = 8
Data = 9

Enter the element to be searched : 6
Element found at node 3


Process returned 27 (0x1B)   execution time : 20.444 s
Press any key to continue.
```

"C:\Users\USER\Downloads\linked list\bin\Debug\linked list.exe"

```
Enter the number of nodes : 5

Enter the data for node 1 : 3
Enter the data for node 2 : 7
Enter the data for node 3 : 6
Enter the data for node 4 : 8
Enter the data for node 5 : 9

Data entered in the list are :
Data = 3
Data = 7
Data = 6
Data = 8
Data = 9

Enter the element to be searched : 2
This element does not exists in linked list.


Process returned 0 (0x0)   execution time : 15.717 s
Press any key to continue.
```

4) Write a program in C to count number of nodes in the circular linked list.

Pseudo code:

1.Begin

2.Initialize the structure and variables

```
3. procedure createList(int n)
   int i, data
   struct node *prevNode, *newNode
   if(n >= 1)
          head = (struct node *)malloc(sizeof(struct node))
          print Enter data
          scanf("%d", &data)
         head->data = data
         head->next = NULL
         prevNode = head
                 for(i=2; i<=n; i++)
                         newNode = (struct node *)malloc(sizeof(struct node))
                          print Enter data
                          scanf("%d", &data)
                         newNode->data = data
                         newNode->next = NULL
                         prevNode->next = newNode
                          prevNode = newNode
                  end for
        prevNode->next = head
        print CIRCULAR LINKED LIST CREATED SUCCESSFULLY
```

```
endif

end procedure

4. procedure displayList()
   struct node *current
   int n = 1
   if(head == NULL)
      printf("List is empty.\n");
   endif
   else
      current = head
      print DATA IN THE LIST
      do
         print Data %d = %d\n", n, current->data
         current = current->next
         n++
         while(current != head)
     enddowhile
  endelse
end procedure
5.procedure countNodes()
          int count=0
          struct node*current
          current=head
          do
                    count++
                    current= current->next
                    while(current!=head)
         enddowhile
        print Total number of nodes
end procedure
6.End
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node * next;
}*head;

void createList(int n);
void displayList();
void countNodes();
```

```
void createList(int n)
{
   int i, data;
   struct node *prevNode, *newNode;

   if(n >= 1)
   {
      head = (struct node *)malloc(sizeof(struct node));

      printf("Enter data of 1 node: ");
      scanf("%d", &data);

      head->data = data;
      head->next = NULL;

      prevNode = head;
      for(i=2; i<=n; i++)
      {
         newNode = (struct node *)malloc(sizeof(struct node));

         printf("Enter data of %d node: ", i);
         scanf("%d", &data);

         newNode->data = data;
         newNode->next = NULL;
         prevNode->next = newNode;
         prevNode = newNode;
      }
      prevNode->next = head;

      printf("\nCIRCULAR LINKED LIST CREATED SUCCESSFULLY\n");
   }
}

void displayList()
{
   struct node *current;
   int n = 1;

   if(head == NULL)
   {
      printf("List is empty.\n");
   }
   else
   {
```

```
    current = head;
    printf("DATA IN THE LIST:\n");

    do {
      printf("Data %d = %d\n", n, current->data);
      current = current->next;
      n++;
    }while(current != head);
  }
}

void countNodes()
{
        int count=0;
        struct node*current;
        current=head;
        do{
                count++;
                current= current->next;
        }while(current!=head);
        printf("Total number of nodes is: %d\n", count);
}

int main()
{
        int n;
        printf("Enter the total number of nodes in list: ");
  scanf("%d", &n);
  createList(n);
  displayList();
  printf("\n");
  countNodes();
}
```

Output

"C:\Users\USER\Downloads\linked list\bin\Debug\linked list.exe"

```
Enter the total number of nodes in list: 5
Enter data of 1 node: 24
Enter data of 2 node: 56
Enter data of 3 node: 78
Enter data of 4 node: 90
Enter data of 5 node: 22

CIRCULAR LINKED LIST CREATED SUCCESSFULLY
DATA IN THE LIST:
Data 1 = 24
Data 2 = 56
Data 3 = 78
Data 4 = 90
Data 5 = 22

Total number of nodes is: 5

Process returned 0 (0x0)   execution time : 15.471 s
Press any key to continue.
```

5) Write a C program to count odd numbers in the singly linked list.

Pseudo code:

1.Begin

2.Initialize the structure and variables

3.procedure createlist()

```
    if(n >= 1)
        head = (struct node *)malloc(sizeof(struct node))
         print Enter data
         scanf("%d", &data)
         head->data = data
         head->next = NULL
         prevNode = head
        for(i=2; i<=n; i++)
                newNode = (struct node *)malloc(sizeof(struct node))
                 print Enter data
                scanf("%d", &data)
                 newNode->data = data
                 newNode->next = NULL
                prevNode->next = newNode
                prevNode = newNode
        Endfor
     Print linked list
    Endif
  End procedure
```

4. procedure displayList()

```
    struct node *current
    int n = 1
    if(head == NULL)
          Print list is empty
    Endif
      else
                    current = head
                    print DATA IN THE LIST
              do
                    print Data %d = %d\n", n, current->data
                    current = current->next
                     n increment
                    while(current != NULL)
             enddowhile
      endelse
 end procedure
5. procedure countOdd()
          count=0
          struct node*current
          current=head
          do
                     tempData = current->data
                    if (tempData % 2 == 1)
                              count increment
                              current=current->next
                    endif
                    while(current != NULL)
          enddowhile
display the odd elements
end procedure
6.End
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node * next;
}*head;

void createList(int n);
void displayList();
void countOdd();

void createList(int n)
```

```
{
   int i, data;
   struct node *prevNode, *newNode;

   if(n >= 1)
   {
      head = (struct node *)malloc(sizeof(struct node));

      printf("Enter data of 1 node: ");
      scanf("%d", &data);

      head->data = data;
      head->next = NULL;

      prevNode = head;
      for(i=2; i<=n; i++)
      {
         newNode = (struct node *)malloc(sizeof(struct node));

         printf("Enter data of %d node: ", i);
         scanf("%d", &data);

         newNode->data = data;
         newNode->next = NULL;
         prevNode->next = newNode;
         prevNode = newNode;
      }
      printf("\nLINKED LIST CREATED SUCCESSFULLY\n");
   }
}

void displayList()
{
   struct node *current;
   int n = 1;

   if(head == NULL)
   {
      printf("List is empty.\n");
   }
   else
   {
      current = head;
      printf("DATA IN THE LIST:\n");
```

```
        do {
            printf("Data %d = %d\n", n, current->data);
            current = current->next;
            n++;
        }while(current != NULL);
    }
}

void countOdd()
{
            int count=0;
            struct node*current;
            current=head;
            do {
        int tempData = current->data;
        if (tempData % 2 == 1)
                count++;
        current=current->next;
        }while(current != NULL);
    printf("Total odd numbers are: %d",count);
}

int main()
{
            int n;
            printf("Enter the total number of nodes in list: ");
    scanf("%d", &n);
    createList(n);
    displayList();
    printf("\n");
    countOdd();
}
```

Output

"C:\Users\USER\Downloads\linked list\bin\Debug\linked list.exe"

```
Enter the total number of nodes in list: 5
Enter data of 1 node: 46
Enter data of 2 node: 76
Enter data of 3 node: 89
Enter data of 4 node: 78
Enter data of 5 node: 26

LINKED LIST CREATED SUCCESSFULLY
DATA IN THE LIST:
Data 1 = 46
Data 2 = 76
Data 3 = 89
Data 4 = 78
Data 5 = 26

Total odd numbers are: 1
Process returned 0 (0x0)   execution time : 14.274 s
Press any key to continue.
```