**Name: Aditi Kohale**

**Course: C, DSA and C++**

**Topic: Linked List – Assignment 2**

**Q. Find the first occurrence of a number:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Demo

{

int data;

struct Demo \*next;

};

struct Demo \*head=NULL;

int first\_occurence(int num)

{

if(head==NULL)

{

return -1;

}

else

{

int count=0;

int flag=0;

struct Demo \*temp=head;

while(temp!=NULL)

{

count++;

if(temp->data==num)

{

flag=1;

break;

}

temp=temp->next;

}

if(flag==0)

{

return -1;

}

else

{

return count;

}

}

}

void printLL()

{

printf("The Linked list is:\n");

struct Demo \*temp=head;

while(temp!=NULL)

{

printf("%d->",temp->data);

temp=temp->next;

}

printf("\n");

}

struct Demo \*createNode()

{

struct Demo \*newNode=(struct Demo\*)malloc(sizeof(struct Demo));

printf("Enter data:\n");

scanf("%d",&newNode->data);

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct Demo \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct Demo \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void main()

{

int count;

printf("Enter the no. of nodes:\n");

scanf("%d",&count);

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

int num;

printf("Enter the data you want to search:\n");

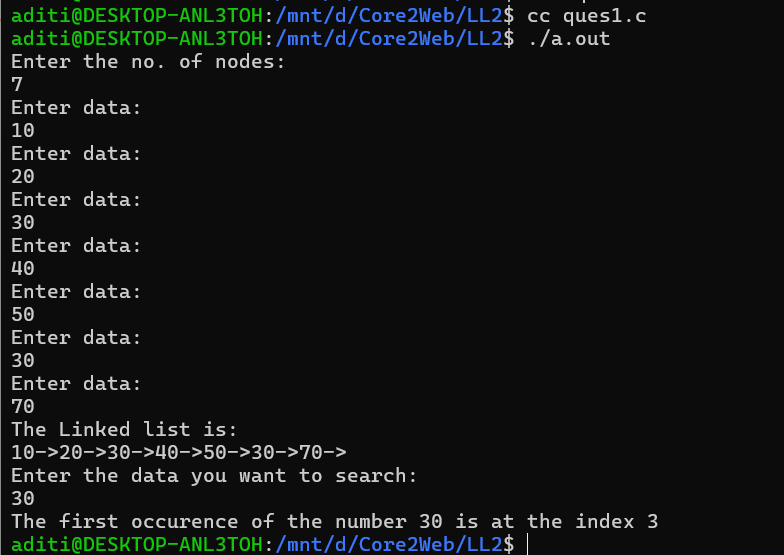
scanf("%d",&num);

int occ=first\_occurence(num);

printf("The first occurence of the number %d is at the index %d\n",num,occ);

}

**Output:**

****

**Q.2. To find the second last occurrence of the number:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Demo

{

int data;

struct Demo \*next;

};

struct Demo \*head=NULL;

void secondLast\_occ(int num)

{

if(head==NULL)

{

printf("Linked list empty\n");

}

else

{

struct Demo \*temp=head;

int count=1;

int last=-1;

int secondLast=-1;

while(temp!=NULL)

{

if(temp->data==num)

{

secondLast=last;

last=count;

}

temp=temp->next;

count++;

}

if(secondLast!=-1)

{

printf("The second last occurence of %d is found at %d\n",num,secondLast);

}

else if(last!=-1)

{

printf("Not second last but last occurence of %d was found at %d\n",num,last);

}

else

{

printf("No occurence found\n");

}

}

}

void printLL()

{

printf("The Linked list is:\n");

struct Demo \*temp=head;

while(temp!=NULL)

{

printf("%d->",temp->data);

temp=temp->next;

}

printf("\n");

}

struct Demo \*createNode()

{

struct Demo \*newNode=(struct Demo\*)malloc(sizeof(struct Demo));

printf("Enter data:\n");

scanf("%d",&newNode->data);

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct Demo \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct Demo \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void main()

{

int count;

printf("Enter the no. of nodes:\n");

scanf("%d",&count);

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

int num;

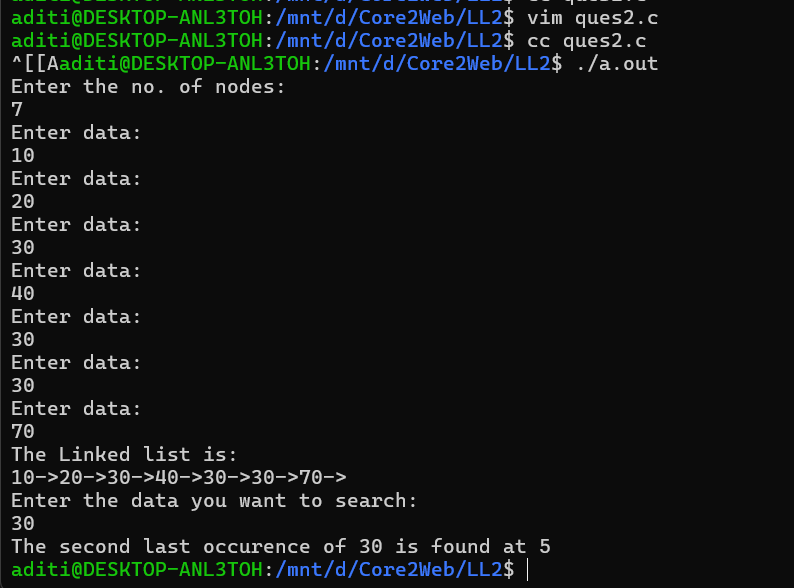
printf("Enter the data you want to search:\n");

scanf("%d",&num);

secondLast\_occ(num);

}

**Output:**

****

**Q.3. WAP that searches the occurrences of the data in the linked list:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Demo

{

int data;

struct Demo \*next;

};

struct Demo \*head=NULL;

int occurence(int num)

{

if(head==NULL)

{

return -1;

}

else

{

int count=0;

int flag=0;

struct Demo \*temp=head;

while(temp!=NULL)

{

if(temp->data==num)

{

count++;

flag=1;

}

temp=temp->next;

}

if(flag==0)

{

return -1;

}

else

{

return count;

}

}

}

void printLL()

{

printf("The Linked list is:\n");

struct Demo \*temp=head;

while(temp!=NULL)

{

printf("%d->",temp->data);

temp=temp->next;

}

printf("\n");

}

struct Demo \*createNode()

{

struct Demo \*newNode=(struct Demo\*)malloc(sizeof(struct Demo));

printf("Enter data:\n");

scanf("%d",&newNode->data);

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct Demo \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct Demo \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void main()

{

int count;

printf("Enter the no. of nodes:\n");

scanf("%d",&count);

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

int num;

printf("Enter the data you want to search:\n");

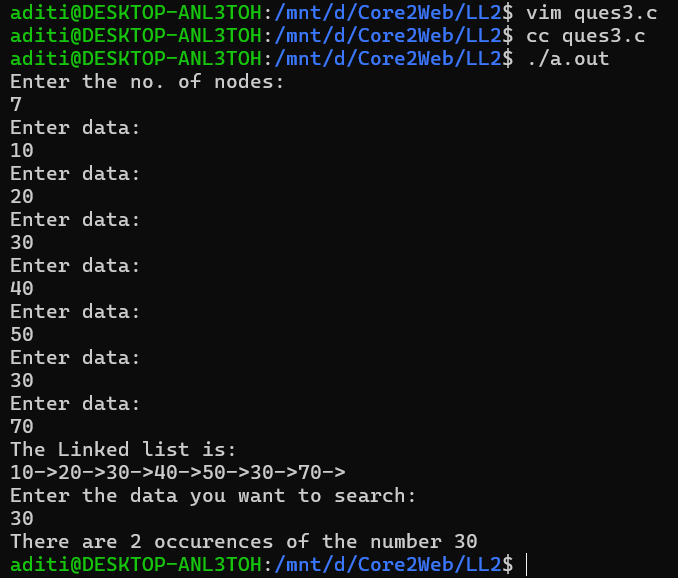
scanf("%d",&num);

int occ=occurence(num);

printf("There are %d occurences of the number %d\n",occ,num);

}

**Output:**

****

**Q.4. WAP that adds the digits of the data at each node:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Demo

{

int data;

struct Demo \*next;

};

struct Demo \*head=NULL;

void addDigits()

{

if(head==NULL)

{

printf("Linked list is empty\n");

}

else

{

struct Demo \*temp=head;

while(temp!=NULL)

{

int org=temp->data;

int last=0;

int sum=0;

while(org!=0)

{

last=org%10;

sum=sum+last;

org=org/10;

}

temp->data=sum;

temp=temp->next;

}

}

}

void printLL()

{

printf("The Linked list is:\n");

struct Demo \*temp=head;

while(temp!=NULL)

{

printf("%d->",temp->data);

temp=temp->next;

}

printf("\n");

}

struct Demo \*createNode()

{

struct Demo \*newNode=(struct Demo\*)malloc(sizeof(struct Demo));

printf("Enter data:\n");

scanf("%d",&newNode->data);

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct Demo \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct Demo \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void main()

{

int count;

printf("Enter the no. of nodes:\n");

scanf("%d",&count);

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

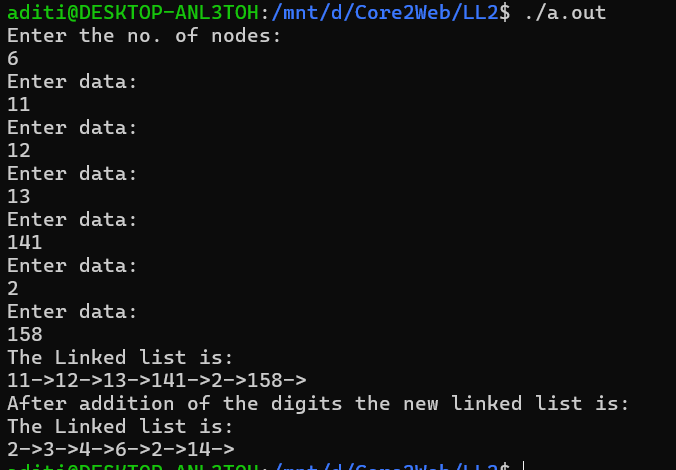
addDigits();

printf("After addition of the digits the new linked list is:\n");

printLL();

}

**Output:**

****

**Q.5. WAP that seaches all the palindrome from the linked list and prints the position of the palindrome:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Demo

{

int data;

struct Demo \*next;

};

struct Demo \*head=NULL;

void checkPallindrome()

{

if(head==NULL)

{

printf("Linked list is empty\n");

}

else

{

int count=1;

struct Demo \*temp=head;

while(temp!=NULL)

{

int org=temp->data;

int last=0;

int rev=0;

while(org!=0)

{

last=org%10;

rev=(rev\*10)+last;

org=org/10;

}

if(rev==temp->data)

{

printf("Pallindrome found at index %d\n",count);

}

temp=temp->next;

count++;

}

}

}

void printLL()

{

printf("The Linked list is:\n");

struct Demo \*temp=head;

while(temp!=NULL)

{

printf("%d->",temp->data);

temp=temp->next;

}

printf("\n");

}

struct Demo \*createNode()

{

struct Demo \*newNode=(struct Demo\*)malloc(sizeof(struct Demo));

printf("Enter data:\n");

scanf("%d",&newNode->data);

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct Demo \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct Demo \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void main()

{

int count;

printf("Enter the no. of nodes:\n");

scanf("%d",&count);

for(int i=1;i<=count;i++)

{

addNode();

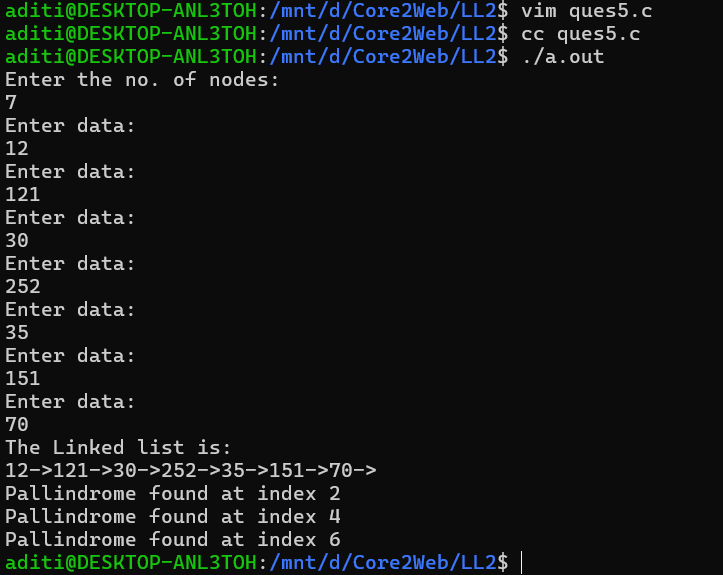
}

printLL();

checkPallindrome();

}

**Output:**

****

**Q.6. WAP that accepts a singly linear linked list from the user. Take a number from the user and print the data of that length.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char str[20];

struct node \*next;

};

struct node \*head=NULL;

int mystrlen(char \*str)

{

int len=0;

while(\*str!='\0')

{

len++;

str++;

}

return len;

}

void count\_char(int num)

{

if(head==NULL)

{

printf("Linked list is empty\n");

}

else

{

struct node \*temp=head;

while(temp!=NULL)

{

int len=mystrlen(temp->str);

if(len==num)

{

printf("%s\n",temp->str);

}

temp=temp->next;

}

}

}

struct node \*createNode()

{

struct node \*newNode=(struct node\*)malloc(sizeof(struct node));

printf("Enter name:\n");

fgets(newNode->str,15,stdin);

int len=mystrlen(newNode->str);

if(newNode->str[len-1]=='\n')

{

newNode->str[len-1]='\0';

}

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct node \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct node \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void printLL()

{

struct node \*temp=head;

while(temp!=NULL)

{

printf("|%s|-->",temp->str);

temp=temp->next;

}

printf("\n");

}

void main()

{

int count;

printf("Enter the number of nodes:\n");

scanf("%d",&count);

getchar();

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

printf("Enter a number:\n");

int num;

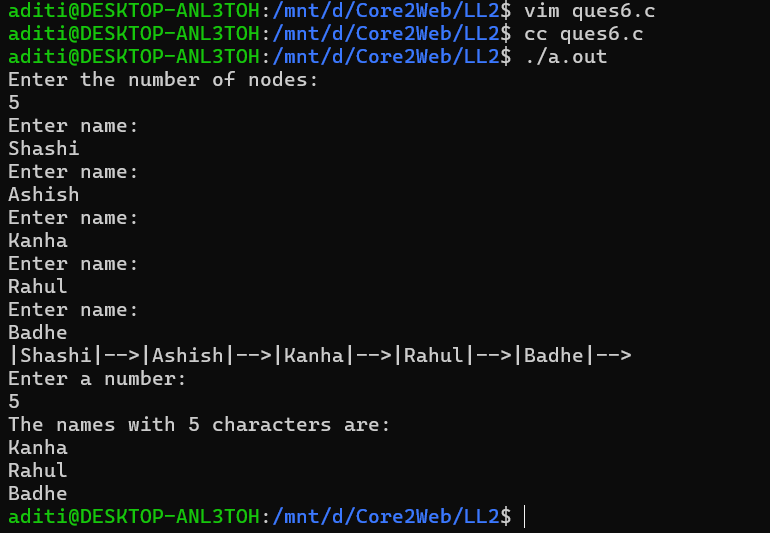
scanf("%d",&num);

printf("The names with %d characters are:\n",num);

count\_char(num);

}

**Output:**

****

**Q.7. Reverse the data elements in the above example:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char str[20];

struct node \*next;

};

struct node \*head=NULL;

int mystrlen(char \*str)

{

int len=0;

while(\*str!='\0')

{

len++;

str++;

}

return len;

}

char \*mystrrev(char \*str)

{

char \*start=str;

char \*temp=str;

while(\*temp!='\0')

{

temp++;

}

temp--;

while(start<temp)

{

char tempvar=\*start;

\*start=\*temp;

\*temp=tempvar;

start++;

temp--;

}

return str;

}

void reverse()

{

if(head==NULL)

{

printf("Linked list is empty\n");

}

else

{

struct node \*temp=head;

while(temp!=NULL)

{

strcpy(temp->str,mystrrev(temp->str));

temp=temp->next;

}

}

}

struct node \*createNode()

{

struct node \*newNode=(struct node\*)malloc(sizeof(struct node));

printf("Enter name:\n");

fgets(newNode->str,15,stdin);

int len=mystrlen(newNode->str);

if(newNode->str[len-1]=='\n')

{

newNode->str[len-1]='\0';

}

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct node \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct node \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void printLL()

{

struct node \*temp=head;

while(temp!=NULL)

{

printf("|%s|-->",temp->str);

temp=temp->next;

}

printf("\n");

}

void main()

{

int count;

printf("Enter the number of nodes:\n");

scanf("%d",&count);

getchar();

for(int i=1;i<=count;i++)

{

addNode();

}

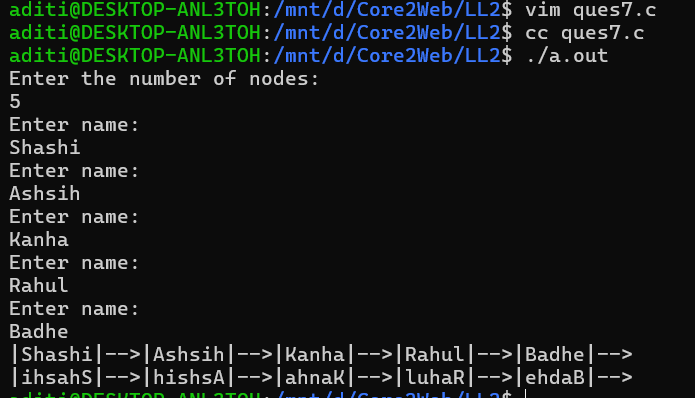
printLL();

reverse();

printLL();

}

**Output:**

****

**Q.8. Keep only the elements in the linked list whose length is equal to the number taken from the user:**

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char str[20];

struct node \*next;

};

struct node \*head=NULL;

int mystrlen(char \*str)

{

int len=0;

while(\*str!='\0')

{

len++;

str++;

}

return len;

}

void limit\_count(int limit)

{

if(head==NULL)

{

printf("Linked list is empty\n");

}

else

{

struct node \*temp=head;

struct node \*prev=NULL;

while(temp!=NULL)

{

int len=mystrlen(temp->str);

if(len!=limit)

{

struct node \*node\_to\_delete=temp;

if(prev==NULL)

{

head=temp->next;

temp=head;

}

else

{

prev->next=temp->next;

temp=temp->next;

}

free(node\_to\_delete);

}

else

{

prev=temp;

temp=temp->next;

}

}

}

}

struct node \*createNode()

{

struct node \*newNode=(struct node\*)malloc(sizeof(struct node));

printf("Enter name:\n");

fgets(newNode->str,15,stdin);

int len=mystrlen(newNode->str);

if(newNode->str[len-1]=='\n')

{

newNode->str[len-1]='\0';

}

newNode->next=NULL;

return newNode;

}

void addNode()

{

struct node \*newNode=createNode();

if(head==NULL)

{

head=newNode;

}

else

{

struct node \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newNode;

}

}

void printLL()

{

struct node \*temp=head;

while(temp!=NULL)

{

printf("|%s|-->",temp->str);

temp=temp->next;

}

printf("\n");

}

void main()

{

int count;

printf("Enter the number of nodes:\n");

scanf("%d",&count);

getchar();

for(int i=1;i<=count;i++)

{

addNode();

}

printLL();

int limit;

printf("Enter the limit:\n");

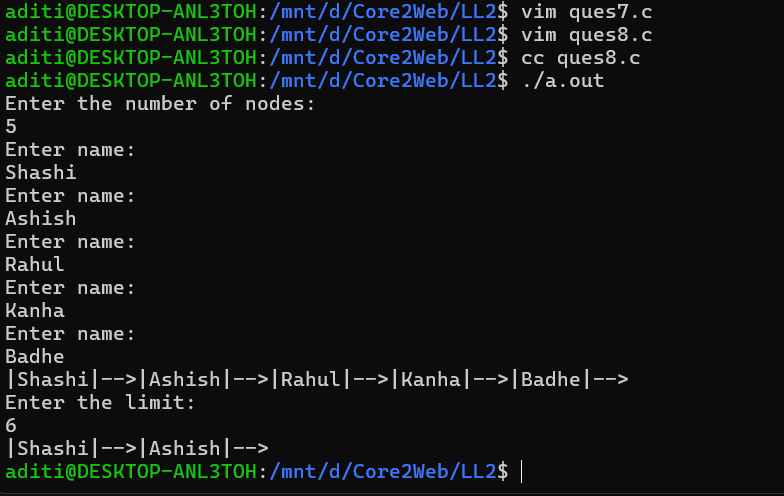
scanf("%d",&limit);

limit\_count(limit);

printLL();

}

**Output:**

****