Name: Aditi Kohale

Course: C, DSA and C++

Topic: Linked List – Assignment 3

Q.1. Count the occurrence of a number:

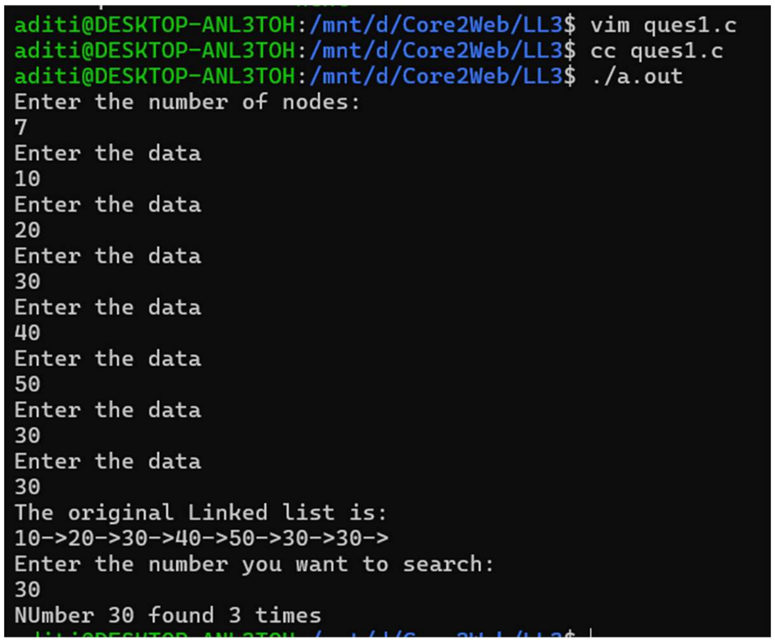
Code:

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

struct node
{
	int data;
	struct node *next;
};

struct node *head=NULL;

int count_occ(int num)
{
	int count=0;
	struct node *temp=head;
	while(temp!=NULL)
	{
		if(temp->data==num)
		{
			count++;
		}
		temp=temp->next;
	}
```

```
        if(count==0)
        {
                return -1;
        }
        else
        {
                return count;
        }
}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        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("%d->",temp->data);
                temp=temp->next;
        }
        printf("\n");
}

void main()
{
        int count;
        printf("Enter the number of nodes:\n");
        scanf("%d",&count);
        for(int i=1;i<=count;i++)
        {
                addNode();
        }
        printf("The original Linked list is:\n");
```

```
	printLL();
	int num;
	printf("Enter the number you want to search:\n");
	scanf("%d",&num);
	int occ=count_occ(num);
	if(occ==-1)
	{
		printf("Number %d not found\n",num);
	}
	else
	{
		printf("NUmber %d found %d times\n",num,occ);
	}
}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques1.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques1.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes:
7
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
40
Enter the data
50
Enter the data
30
Enter the data
30
The original Linked list is:
10->20->30->40->50->30->30->
Enter the number you want to search:
30
NUmber 30 found 3 times
```

Q.2. Take two singly linked list from the user and concat the two:

Code:

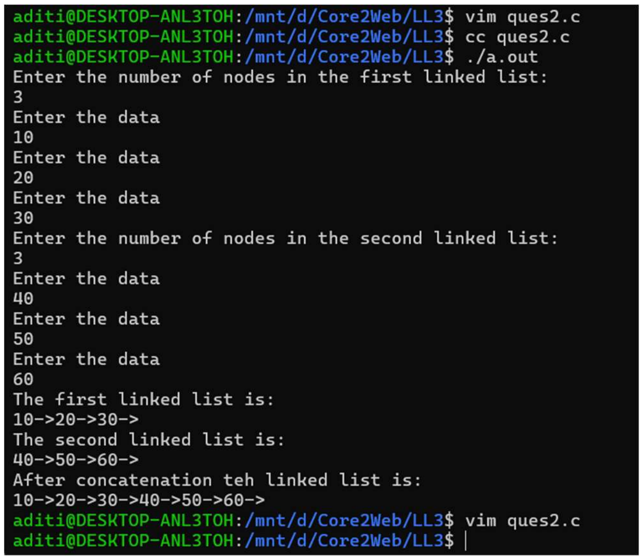
```
#include<stdio.h>
#include<stdlib.h>

struct node
{
	int data;
	struct node *next;
};

struct node *concat(struct node *head1,struct node *head2)
{
	if(head2==NULL && head1 ==NULL)
	{
		printf("Both linked list are empty.\n");
		return NULL;
	}
	else if(head2==NULL)
	{
		printf("The second linked list is empty\n");
		return head1;
	}
	else if(head1==NULL)
	{
		printf("The first linked list is empty\n");
		return head2;
	}
```

```
        else
        {
                struct node *temp1=head1;
                while(temp1->next!=NULL)
                {
                        temp1=temp1->next;
                }
                temp1->next=head2;

                return head1;
        }
}








struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
```

```
{
	struct node *newNode=createNode();
	if(head==NULL)
	{
		head=newNode;
	}
	else
	{
		struct node *temp=head;
		while(temp->next!=NULL)
		{
			temp=temp->next;
		}
		temp->next=newNode;
	}
	return head;
}

void printLL(struct node *head)
{
	struct node *temp=head;
	while(temp!=NULL)
	{
		printf("%d->",temp->data);
		temp=temp->next;
	}
	printf("\n");
}
```

```
void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c1,c2;
        printf("Enter the number of nodes in the first linked list:\n");
        scanf("%d",&c1);
        for(int i=1;i<=c1;i++)
        {
                head1=addNode(head1);
        }
        printf("Enter the number of nodes in the second linked list:\n");
        scanf("%d",&c2);
        for(int i=1;i<=c2;i++)
        {
                head2=addNode(head2);
        }
        printf("The first linked list is:\n");
        printLL(head1);
        printf("The second linked list is:\n");
        printLL(head2);
        head1 = concat(head1,head2);
        printf("After concatenation teh linked list is:\n");
        printLL(head1);

}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques2.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques2.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the first linked list:
3
Enter the data
10
Enter the data
20
Enter the data
30
Enter the number of nodes in the second linked list:
3
Enter the data
40
Enter the data
50
Enter the data
60
The first linked list is:
10->20->30->
The second linked list is:
40->50->60->
After concatenation teh linked list is:
10->20->30->40->50->60->
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques2.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q.3. Concat the first N elements of the source linked list after the destination linked list:

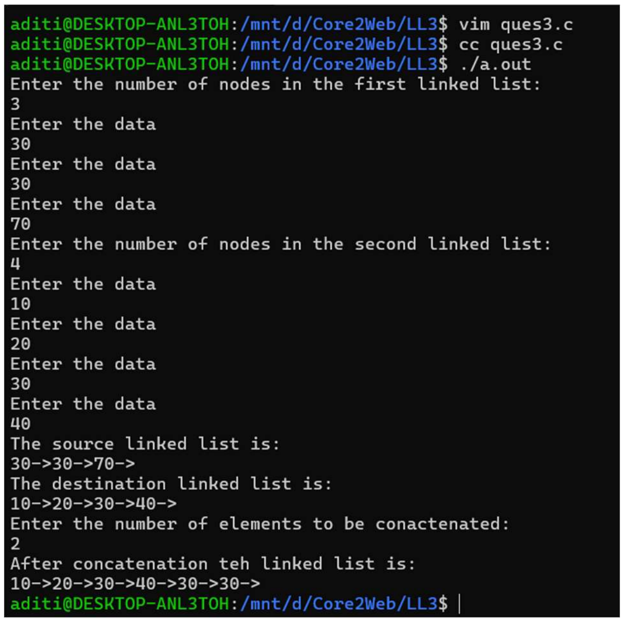
Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

	int data;

	struct node *next;

};


struct node *concat(struct node *head1,struct node *head2,int n)

{

	if(head2==NULL && head1 ==NULL)

	{

		printf("Both linked list are empty.\n");

		return NULL;
	}
	struct node *copyHead=NULL;
	struct node *copyTail=NULL;
	struct node *temp1=head1;
	struct node *temp2=head2;
	int count=0;
	while(temp1!=NULL && count<n)
	{
		struct node *newNode=(struct node*)malloc(sizeof(struct node));
		newNode->data=temp1->data;
		newNode->next=NULL;

		if(copyHead==NULL)
		{
			copyHead=newNode;
			copyTail=newNode;
		}
		else
		{
			copyTail->next=newNode;
			copyTail=newNode;
		}
		count++;
		temp1=temp1->next;
	}
	if(count<n)
	{
		printf("Warning: Nodes present in source LL is less than %d. Therefore only %d nodes
concatenated\n",n,count);
```

```
	}
	if(head2==NULL)
	{
		return copyHead;
	}
	else
	{
		while(temp2->next!=NULL)
		{
			temp2=temp2->next;
		}
		temp2->next=copyHead;
		return head2;
	}
}



struct node *createNode()
{
	struct node *newNode=(struct node*)malloc(sizeof(struct node));
	printf("Enter the data\n");
	scanf("%d",&newNode->data);
	newNode->next=NULL;
	return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        struct node *temp=head;
        while(temp!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("\n");
}

void main()
```

```
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c1,c2;
        printf("Enter the number of nodes in the first linked list:\n");
        scanf("%d",&c1);
        for(int i=1;i<=c1;i++)
        {
                head1=addNode(head1);
        }
        printf("Enter the number of nodes in the second linked list:\n");
        scanf("%d",&c2);
        for(int i=1;i<=c2;i++)
        {
                head2=addNode(head2);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        printf("The destination linked list is:\n");
        printLL(head2);
        int n;
        printf("Enter the number of elements to be conactenated:\n");
        scanf("%d",&n);
        head2 = concat(head1,head2,n);
        printf("After concatenation teh linked list is:\n");
        printLL(head2);


}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques3.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques3.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the first linked list:
3
Enter the data
30
Enter the data
30
Enter the data
70
Enter the number of nodes in the second linked list:
4
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
40
The source linked list is:
30->30->70->
The destination linked list is:
10->20->30->40->
Enter the number of elements to be conactenated:
2
After concatenation teh linked list is:
10->20->30->40->30->30->
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q. 4. Take two singly linked list from the user and concat last n elements from the source list to destination list:

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

	int data;

	struct node *next;

};


int countNode(struct node *head)

{

	if(head==NULL)

	{

		return 0;
```

```
        }
        else
        {
                struct node *temp=head;
                int count=0;
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

struct node *concat(struct node *head1,struct node *head2,int n)
{
        if(head2==NULL && head1 ==NULL)
        {
                printf("Both linked list are empty.\n");
                return NULL;
        }
        struct node *copyHead=NULL;
        struct node *copyTail=NULL;
        struct node *temp1=head1;
        struct node *temp2=head2;
        int count_source=countNode(head1);
        if(count_source<n)
        {
```

```
			printf("Source string has less no. of nodes than given nodes, therefore all the nodes of
source string will be concatenated\n");
			if(head2==NULL)
			{
				printf("Destination list is empty\n");
				return head1;
			}
			else if(count_source==0)
			{
				printf("No nodes in source string\n");
				return head2;
			}
			else
			{
				while(temp2->next!=NULL)
				{
					temp2=temp2->next;
				}
				temp2->next=head1;
				return head2;
			}
		}
		else
		{
			int pos=count_source-n;
			while(pos)
			{
				temp1=temp1->next;
				pos--;
```

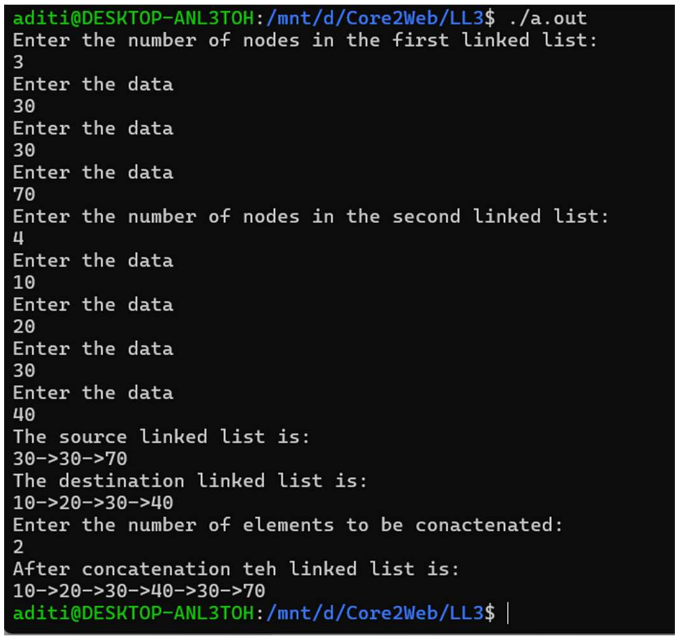
```
	}
while(temp1!=NULL)
{
	struct node *newNode=(struct node*)malloc(sizeof(struct node));
	newNode->data=temp1->data;
	newNode->next=NULL;

	if(copyHead==NULL)
	{
		copyHead=newNode;
		copyTail=newNode;
	}
	else
	{
		copyTail->next=newNode;
		copyTail=newNode;
	}
	temp1=temp1->next;
}
if(head2==NULL)
{
	return copyHead;
}
else
{
	while(temp2->next!=NULL)
	{
		temp2=temp2->next;
	}
```

```
                temp2->next=copyHead;
                return head2;
        }
  }
}



struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                    temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
```

```
int c1,c2;

printf("Enter the number of nodes in the first linked list:\n");

scanf("%d",&c1);

for(int i=1;i<=c1;i++)

{

        head1=addNode(head1);

}

printf("Enter the number of nodes in the second linked list:\n");

scanf("%d",&c2);

for(int i=1;i<=c2;i++)

{

        head2=addNode(head2);

}

printf("The source linked list is:\n");

printLL(head1);

printf("The destination linked list is:\n");

printLL(head2);

int n;

printf("Enter the number of elements to be conactenated:\n");

scanf("%d",&n);

if(n<0)

{

        printf("invalid input\n");

}

else

{

head2 = concat(head1,head2,n);

printf("After concatenation teh linked list is:\n");

printLL(head2);
```

```
          }


}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the first linked list:
3
Enter the data
30
Enter the data
30
Enter the data
70
Enter the number of nodes in the second linked list:
4
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
40
The source linked list is:
30->30->70
The destination linked list is:
10->20->30->40
Enter the number of elements to be conactenated:
2
After concatenation teh linked list is:
10->20->30->40->30->70
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q.5. Take two singly linked list from the user and also take a range from the user and conact the elements from the source linked list within that range after the destination linked list:

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

          int data;

          struct node *next;

};
```

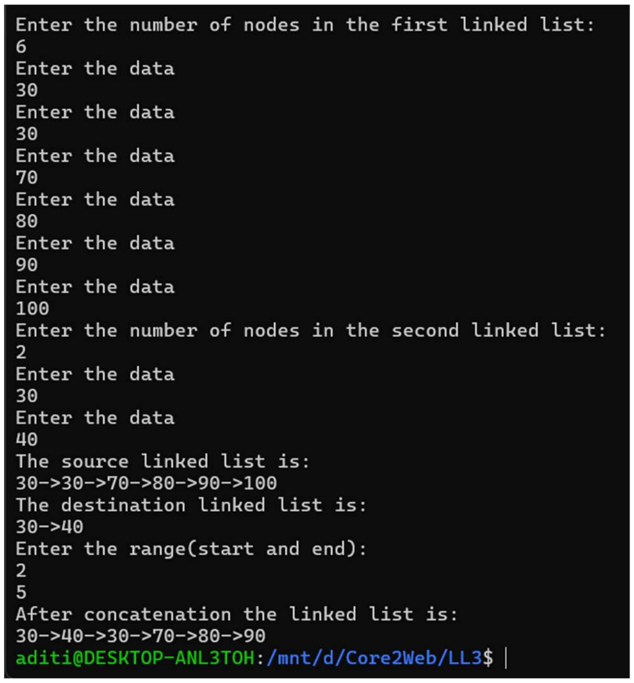
```
int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
		while(temp!=NULL)
		{
			count++;
			temp=temp->next;
		}
		return count;
	}
}

struct node *concat(struct node *head1,struct node *head2,int start, int end)
{
	if(head2==NULL && head1 ==NULL)
	{
		printf("Both linked list are empty.\n");
		return NULL;
	}
	struct node *copyHead=NULL;
	struct node *copyTail=NULL;
```

```
        struct node *temp1=head1;
        struct node *temp2=head2;
        int count_source=countNode(head1);
        int no_of_nodes=end-start+1;
        if(start>count_source && end>count_source)
        {
                printf("Both the start and end are greater than the nodes in the source list\n");
                return head2;
        }
    else if(start>count_source)
        {
                printf("Start is greater than no. of nodes in source list\n");
                return head2;
        }
        else if(end>count_source)
        {
                printf("The end is greater than the no. of nodes in source string, therefore all the nodes
will be concatenated\n");
                if(head2==NULL)
                {
           return head1;
                }
                else
                {
                        while(temp2->next!=NULL)
                        {
                                temp2=temp2->next;
                        }
                        temp2->next=head1;
                        return head2;
                }
        }
        else
        {
                while(start-1)
                {
                        temp1=temp1->next;
                        start--;
                }

        while(no_of_nodes)
        {
                struct node *newNode=(struct node*)malloc(sizeof(struct node));
                newNode->data=temp1->data;
                newNode->next=NULL;

                if(copyHead==NULL)
                {
                        copyHead=newNode;
                        copyTail=newNode;
                }
                else
                {
                        copyTail->next=newNode;
                        copyTail=newNode;
                }
                no_of_nodes--;
                temp1=temp1->next;
```

```
            }
            if(head2==NULL)
            {
                        return copyHead;
            }
            else
            {
                        while(temp2->next!=NULL)
                        {
                                    temp2=temp2->next;
                        }
                        temp2->next=copyHead;
                        return head2;
            }
   }
}



struct node *createNode()
{
            struct node *newNode=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data\n");
            scanf("%d",&newNode->data);
            newNode->next=NULL;
            return newNode;
}
```

```
struct node *addNode(struct node *head)

{

        struct node *newNode=createNode();

        if(head==NULL)

        {

                head=newNode;

        }

        else

        {

                struct node *temp=head;

                while(temp->next!=NULL)

                {

                        temp=temp->next;

                }

                temp->next=newNode;

        }

        return head;

}


void printLL(struct node *head)

{

        if(head==NULL)

        {

                printf("Linked list empty\n");

        }

        else

        {

        struct node *temp=head;

        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c1,c2;
        printf("Enter the number of nodes in the first linked list:\n");
        scanf("%d",&c1);
        for(int i=1;i<=c1;i++)
        {
                head1=addNode(head1);
        }
        printf("Enter the number of nodes in the second linked list:\n");
        scanf("%d",&c2);
        for(int i=1;i<=c2;i++)
        {
                head2=addNode(head2);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        printf("The destination linked list is:\n");
        printLL(head2);
        int start,end;

        printf("Enter the range(start and end):\n");

        scanf("%d",&start);

        scanf("%d",&end);

        if(start<0 || end<0)

        {

                printf("Invalid input\n");

        }

        else

        {

        head2 = concat(head1,head2,start,end);

        printf("After concatenation the linked list is:\n");

        printLL(head2);

        }


}
```

Output:



```
Enter the number of nodes in the first linked list:
6
Enter the data
30
Enter the data
30
Enter the data
70
Enter the data
80
Enter the data
90
Enter the data
100
Enter the number of nodes in the second linked list:
2
Enter the data
30
Enter the data
40
The source linked list is:
30->30->70->80->90->100
The destination linked list is:
30->40
Enter the range(start and end):
2
5
After concatenation the linked list is:
30->40->30->70->80->90
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q.6 WAP that copies that copies first N elements of the source singly linked list to the destination singly linked list:

Code:

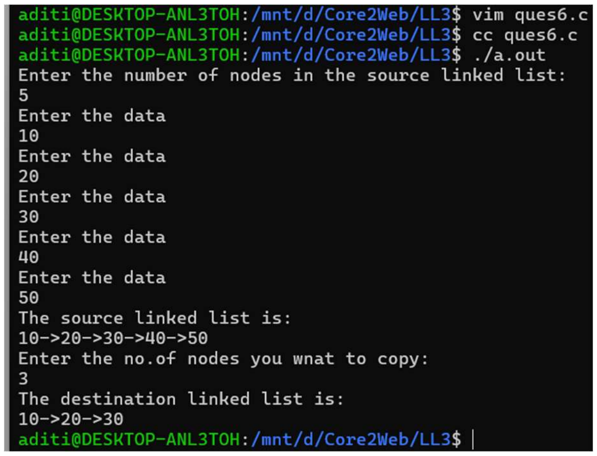
```
#include<stdio.h>

#include<stdlib.h>


struct node

{

	int data;

	struct node *next;

};


int countNode(struct node*);

struct node *addNode(struct node*);

struct node *createNode();

void printLL(struct node*);

struct node *src_to_dest(struct node*, struct node*, int);



int countNode(struct node *head)

{

	if(head==NULL)

	{

		return 0;

	}

	else

	{
```

```
                struct node *temp=head;
                int count=0;
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

struct node *src_to_dest(struct node *head1, struct node *head2, int n)
{
        int count_src=countNode(head1);
        if(n<0)
        {
                printf("Invalid Input\n");
                return NULL;
        }
        if(count_src<n)
        {
                printf("There are less no. of nodes in the source linked list as compared to the give
Number therefore all the nodes are copied from the source list\n");
           n=count_src;
        }
                struct node *copyHead=NULL;
                struct node *copyTail=NULL;
                struct node *temp=head1;
                while(n)
                {
                        struct node *newNode=(struct node*)malloc(sizeof(struct node));
                        newNode->data=temp->data;
                        newNode->next=NULL;

                        if(copyHead==NULL)
                        {
                                copyHead=newNode;
                                copyTail=newNode;
                        }
                        else
                        {
                                copyTail->next=newNode;
                                copyTail=newNode;
                        }
                        n--;
                        temp=temp->next;
                }
                return copyHead;

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
```

```
}

struct node *addNode(struct node *head)
{
	struct node *newNode=createNode();
	if(head==NULL)
	{
		head=newNode;
	}
	else
	{
		struct node *temp=head;
		while(temp->next!=NULL)
		{
			temp=temp->next;
		}
		temp->next=newNode;
	}
	return head;
}

void printLL(struct node *head)
{
	if(head==NULL)
	{
		printf("Linked list empty\n");
	}
	else
	{
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c;
        printf("Enter the number of nodes in the source linked list:\n");
        scanf("%d",&c);
        for(int i=1;i<=c;i++)
        {
                head1=addNode(head1);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        printf("Enter the no.of nodes you wnat to copy:\n");
        int n;
        scanf("%d",&n);
        head2=src_to_dest(head1,head2,n);
        printf("The destination linked list is:\n");
        printLL(head2);
```

}

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques6.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques6.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
5
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
40
Enter the data
50
The source linked list is:
10->20->30->40->50
Enter the no.of nodes you wnat to copy:
3
The destination linked list is:
10->20->30
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q.7. WAP that copies the last n elements from the source linked list to the destination linked list:

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node *next;

};


int countNode(struct node*);

struct node *addNode(struct node*);

struct node *createNode();

void printLL(struct node*);
```

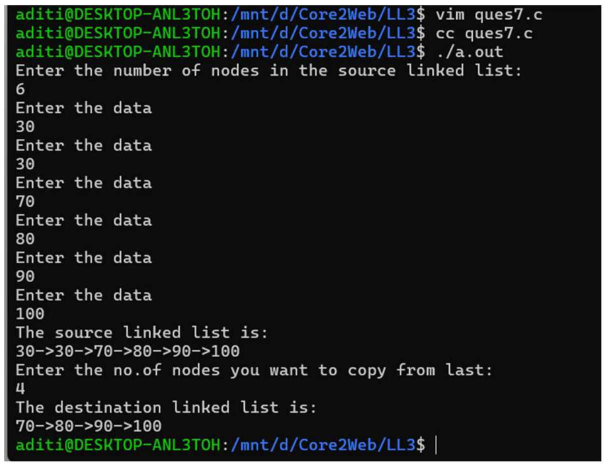
```
struct node *src_to_dest(struct node*, struct node*, int);



int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
		while(temp!=NULL)
		{
			count++;
			temp=temp->next;
		}
		return count;
	}
}

struct node *src_to_dest(struct node *head1, struct node *head2, int n)
{
	int count_src=countNode(head1);
	if(n<0)
	{
		printf("Invalid Input\n");
		return NULL;
```

```
            }
            if(count_src<n)
            {
                        printf("There are less no. of nodes in the source linked list as compared to the give
Number therefore all the nodes are copied from the source list\n");
                n=count_src;
            }
                        struct node *copyHead=NULL;
                        struct node *copyTail=NULL;
                        struct node *temp=head1;
                        int pos=count_src-n;
                        while(pos)
                        {
                                    temp=temp->next;
                                    pos--;
                        }
                        while(temp!=NULL)
                        {
                                    struct node *newNode=(struct node*)malloc(sizeof(struct node));
                                    newNode->data=temp->data;
                                    newNode->next=NULL;

                                    if(copyHead==NULL)
                                    {
                                                copyHead=newNode;
                                                copyTail=newNode;
                                    }
                                    else
                                    {
```

```
                    copyTail->next=newNode;
                    copyTail=newNode;
                }
                n--;
                temp=temp->next;
            }
            return copyHead;

}

struct node *createNode()
{
    struct node *newNode=(struct node*)malloc(sizeof(struct node));
    printf("Enter the data\n");
    scanf("%d",&newNode->data);
    newNode->next=NULL;
    return newNode;
}

struct node *addNode(struct node *head)
{
    struct node *newNode=createNode();
    if(head==NULL)
    {
        head=newNode;
    }
    else
    {
        struct node *temp=head;
```

```
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;

        struct node *head2=NULL;

        int c;

        printf("Enter the number of nodes in the source linked list:\n");

        scanf("%d",&c);

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

        {

                head1=addNode(head1);

        }

        printf("The source linked list is:\n");

        printLL(head1);

        printf("Enter the no.of nodes you want to copy from last:\n");

        int n;

        scanf("%d",&n);

        head2=src_to_dest(head1,head2,n);

        printf("The destination linked list is:\n");

        printLL(head2);


}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques7.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques7.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
6
Enter the data
30
Enter the data
30
Enter the data
70
Enter the data
80
Enter the data
90
Enter the data
100
The source linked list is:
30->30->70->80->90->100
Enter the no.of nodes you want to copy from last:
4
The destination linked list is:
70->80->90->100
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ 
```

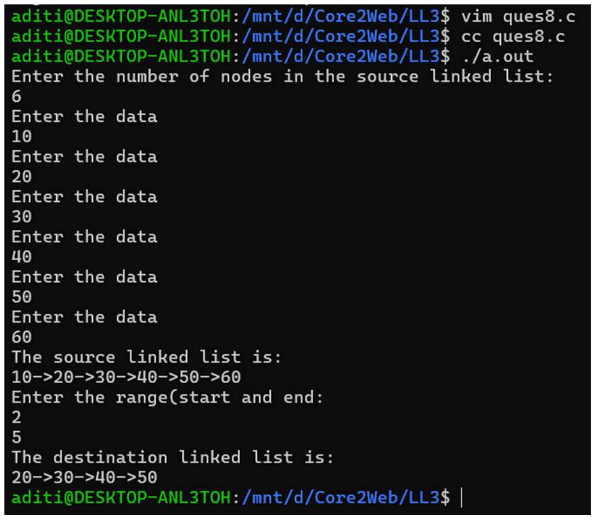
Q.8. WAP that copies the content of the source linked list to the destination linked list which lies between the particular range accepted by the user:

Code:

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

struct node
{
	int data;
	struct node *next;
};

int countNode(struct node*);
struct node *addNode(struct node*);
struct node *createNode();
void printLL(struct node*);
struct node *src_to_dest(struct node*, struct node*, int, int);


int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
```

```
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

struct node *src_to_dest(struct node *head1, struct node *head2, int start, int end)
{
        int count_src=countNode(head1);
        if(start<0 || end<0)
        {
                printf("Invalid Input\n");
                return NULL;
        }
        if(start>count_src && end>count_src)
        {
                printf("Both start and end are greater than the source list count\n");
                return NULL;
        }
        if(start>count_src)
        {
                printf("The start is greater than the node count in source list\n");
                return NULL;
        }
        if(end>count_src)
        {
                printf("The end is greater than the no. of nodes in source list therefore, copying all the
content\n");
                end=count_src;
        }

                struct node *copyHead=NULL;
                struct node *copyTail=NULL;
                struct node *temp=head1;
                int no_of_nodes=end-start+1;
                while(start-1)
                {
                        temp=temp->next;
                        start--;
                }
                while(no_of_nodes)
                {
                        struct node *newNode=(struct node*)malloc(sizeof(struct node));
                        newNode->data=temp->data;
                        newNode->next=NULL;

                        if(copyHead==NULL)
                        {
                                copyHead=newNode;
                                copyTail=newNode;
                        }
                        else
                        {
                                copyTail->next=newNode;
                                copyTail=newNode;
                        }
                        no_of_nodes--;
                        temp=temp->next;
                }
                return copyHead;

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c;

        printf("Enter the number of nodes in the source linked list:\n");

        scanf("%d",&c);

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

        {

                head1=addNode(head1);

        }

        printf("The source linked list is:\n");

        printLL(head1);

        printf("Enter the range(start and end:\n");

        int start,end;

        scanf("%d",&start);

        scanf("%d",&end);

        head2=src_to_dest(head1,head2,start,end);

        printf("The destination linked list is:\n");

        printLL(head2);


}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques8.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques8.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
6
Enter the data
10
Enter the data
20
Enter the data
30
Enter the data
40
Enter the data
50
Enter the data
60
The source linked list is:
10->20->30->40->50->60
Enter the range(start and end:
2
5
The destination linked list is:
20->30->40->50
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$
```

Q.9. WAP that copies the alternate contents of the source linked list to the destination linked list:

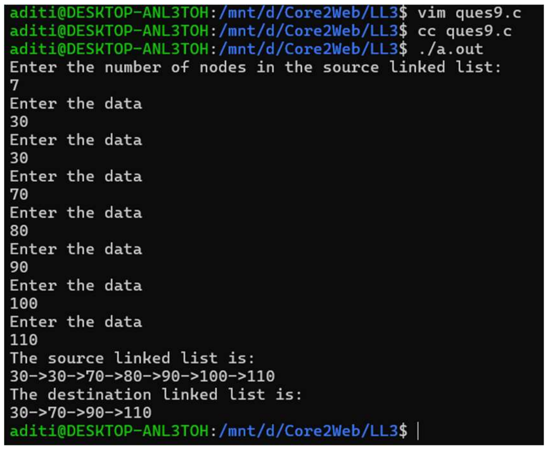
Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node *next;

};


int countNode(struct node*);

struct node *addNode(struct node*);

struct node *createNode();

void printLL(struct node*);

struct node *src_to_dest(struct node*, struct node*);



int countNode(struct node *head)

{

        if(head==NULL)

        {

                return 0;

        }

        else

        {

                struct node *temp=head;

                int count=0;
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

struct node *src_to_dest(struct node *head1, struct node *head2)
{
        int count_src=countNode(head1);
        if(head1==NULL)
        {
                printf("Linked List empty\n");
                return NULL;
        }
                struct node *copyHead=NULL;
                struct node *copyTail=NULL;
                struct node *temp=head1;
                while(temp->next!=NULL)
                {
                        struct node *newNode=(struct node*)malloc(sizeof(struct node));
                        newNode->data=temp->data;
                        newNode->next=NULL;

                        if(copyHead==NULL)
                        {
                                copyHead=newNode;
```

```
                    copyTail=newNode;
                }
                else
                {
                    copyTail->next=newNode;
                    copyTail=newNode;
                }
                temp=temp->next->next;
            }
            copyTail->next=temp;
            return copyHead;

}

struct node *createNode()
{
    struct node *newNode=(struct node*)malloc(sizeof(struct node));
    printf("Enter the data\n");
    scanf("%d",&newNode->data);
    newNode->next=NULL;
    return newNode;
}

struct node *addNode(struct node *head)
{
    struct node *newNode=createNode();
    if(head==NULL)
    {
        head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
```

```
}


void main()
{
	struct node *head1=NULL;
	struct node *head2=NULL;
	int c;
	printf("Enter the number of nodes in the source linked list:\n");
	scanf("%d",&c);
	for(int i=1;i<=c;i++)
	{
		head1=addNode(head1);
	}
	printf("The source linked list is:\n");
	printLL(head1);
	head2=src_to_dest(head1,head2);
	printf("The destination linked list is:\n");
	printLL(head2);


}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques9.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques9.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
7
Enter the data
30
Enter the data
30
Enter the data
70
Enter the data
80
Enter the data
90
Enter the data
100
Enter the data
110
The source linked list is:
30->30->70->80->90->100->110
The destination linked list is:
30->70->90->110
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ 
```

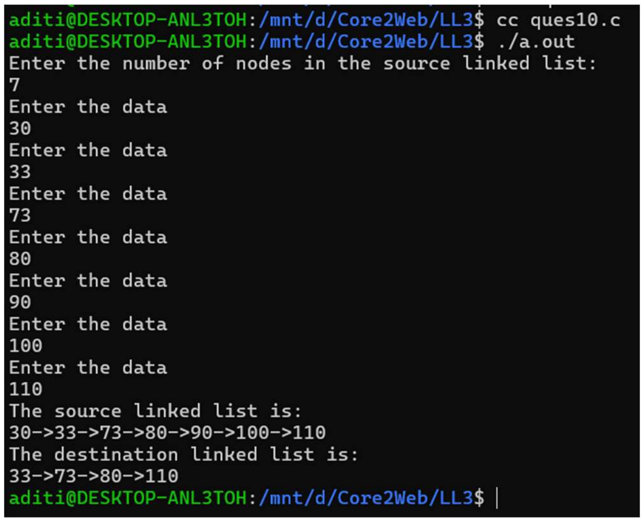
Q.10. WAP that copies the contents of a source singly linked list to destination list whose addition of digits makes up even No.:

Code:

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

struct node
{
	int data;
	struct node *next;
};

int countNode(struct node*);
struct node *addNode(struct node*);
struct node *createNode();
void printLL(struct node*);
struct node *src_to_dest(struct node*, struct node*);


int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
```

```
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

struct node *src_to_dest(struct node *head1, struct node *head2)
{
        int count_src=countNode(head1);
        if(head1==NULL)
        {
                printf("Linked List empty\n");
                return NULL;
        }
                struct node *copyHead=NULL;
                struct node *copyTail=NULL;
                struct node *temp=head1;
                while(temp!=NULL)
                {
                        int data=temp->data;
                        int sum=0;
                        int last_digit=0;
                        while(data!=0)
                        {
                                last_digit=data%10;
                                sum=sum+last_digit;
                                data=data/10;
                        }
                        if(sum%2==0)
                        {
                        struct node *newNode=(struct node*)malloc(sizeof(struct node));
                        newNode->data=temp->data;
                        newNode->next=NULL;

                        if(copyHead==NULL)
                        {
                                copyHead=newNode;
                                copyTail=newNode;
                        }
                        else
                        {
                                copyTail->next=newNode;
                                copyTail=newNode;
                        }
                      }
                        temp=temp->next;
                }
                copyTail->next=temp;
                return copyHead;

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c;
        printf("Enter the number of nodes in the source linked list:\n");
        scanf("%d",&c);
        for(int i=1;i<=c;i++)
        {
                head1=addNode(head1);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        head2=src_to_dest(head1,head2);
        printf("The destination linked list is:\n");
        printLL(head2);

}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques10.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
7
Enter the data
30
Enter the data
33
Enter the data
73
Enter the data
80
Enter the data
90
Enter the data
100
Enter the data
110
The source linked list is:
30->33->73->80->90->100->110
The destination linked list is:
33->73->80->110
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ 
```

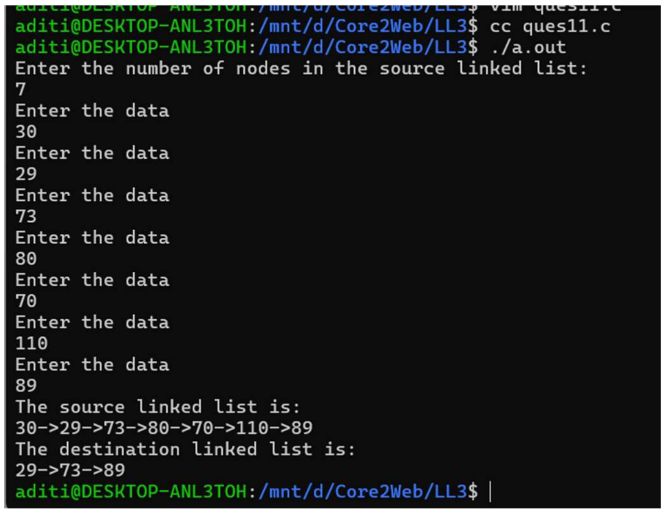
Q.11. Wap to copy the data of the source singly linked list to the destination singly linked list such that the contents should be prime number:

Code:

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

struct node
{
        int data;
        struct node *next;
};

int countNode(struct node*);
struct node *addNode(struct node*);
struct node *createNode();
void printLL(struct node*);
struct node *src_to_dest(struct node*, struct node*);


int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
		while(temp!=NULL)
		{
			count++;
			temp=temp->next;
		}
		return count;
	}
}

struct node *src_to_dest(struct node *head1, struct node *head2)
{
	int count_src=countNode(head1);
	if(head1==NULL)
	{
        printf("Linked List empty\n");
        return NULL;
}
        struct node *copyHead=NULL;
        struct node *copyTail=NULL;
        struct node *temp=head1;
        int flag=0;
        while(temp!=NULL)
        {
                int data=temp->data;
                int flag=1;
                if(data<2)
                {
                        flag=0;
                }
                else
                {
  for(int i=2;i<data;i++)
                        {
                                if(data%i==0)
                                {
                                        flag=0;
                                        break;
                                }
                        }
                }

                if(flag)
                {
                        struct node *newNode=(struct node*)malloc(sizeof(struct node));
                        newNode->data=temp->data;
                        newNode->next=NULL;

                        if(copyHead==NULL)
                        {
                                copyHead=newNode;
                                copyTail=newNode;
                        }
                        else
                        {
                                copyTail->next=newNode;
                                copyTail=newNode;
                        }
                    }
                        temp=temp->next;
                }
                return copyHead;

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}
```

```
struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
                printf("Linked list empty\n");
        }
        else
        {
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
        }
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c;
        printf("Enter the number of nodes in the source linked list:\n");
        scanf("%d",&c);
        for(int i=1;i<=c;i++)
        {
                head1=addNode(head1);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        head2=src_to_dest(head1,head2);
        printf("The destination linked list is:\n");
        printLL(head2);

}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques11.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques11.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
7
Enter the data
30
Enter the data
29
Enter the data
73
Enter the data
80
Enter the data
70
Enter the data
110
Enter the data
89
The source linked list is:
30->29->73->80->70->110->89
The destination linked list is:
29->73->89
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$
```

Q.12. WAP that copies the contents of the source singly linked list to the destination singly linked list such that the sum of the digits of the data is a prime number:

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node *next;

};


int countNode(struct node*);

struct node *addNode(struct node*);

struct node *createNode();

void printLL(struct node*);

struct node *src_to_dest(struct node*, struct node*);
```

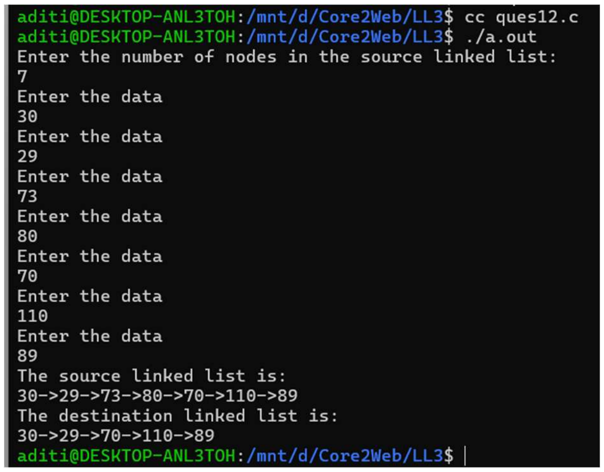
```
int countNode(struct node *head)
{
	if(head==NULL)
	{
		return 0;
	}
	else
	{
		struct node *temp=head;
		int count=0;
		while(temp!=NULL)
		{
			count++;
			temp=temp->next;
		}
		return count;
	}
}

struct node *src_to_dest(struct node *head1, struct node *head2)
{
	int count_src=countNode(head1);
	if(head1==NULL)
	{
		printf("Linked List empty\n");
		return NULL;
	}
```

```
struct node *copyHead=NULL;
struct node *copyTail=NULL;
struct node *temp=head1;
int flag=0;
while(temp!=NULL)
{
        int data=temp->data;
        int flag=1;
        int data_sum=0;
        int last_digit=0;
        while(data!=0)
        {
                last_digit=data%10;
                data_sum+=last_digit;
                data=data/10;
        }
        if(data_sum<2)
        {
                flag=0;
        }
        else
        {
  for(int i=2;i<data_sum;i++)
                {
                        if(data_sum%i==0)
                        {
                                flag=0;
                                break;
                        }
                        }
                }

                if(flag)
                {
                struct node *newNode=(struct node*)malloc(sizeof(struct node));
                newNode->data=temp->data;
                newNode->next=NULL;

                if(copyHead==NULL)
                {
                        copyHead=newNode;
                        copyTail=newNode;
                }
                else
                {
                        copyTail->next=newNode;
                        copyTail=newNode;
                }
              }
                temp=temp->next;
        }
        return copyHead;

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        if(head==NULL)
        {
```

```
		printf("Linked list empty\n");
	}
	else
	{
	struct node *temp=head;
	while(temp->next!=NULL)
	{
		printf("%d->",temp->data);
		temp=temp->next;
	}
	printf("%d\n",temp->data);
	}
}

void main()
{
	struct node *head1=NULL;
	struct node *head2=NULL;
	int c;
	printf("Enter the number of nodes in the source linked list:\n");
	scanf("%d",&c);
	for(int i=1;i<=c;i++)
	{
		head1=addNode(head1);
	}
	printf("The source linked list is:\n");
	printLL(head1);
	head2=src_to_dest(head1,head2);
	printf("The destination linked list is:\n");
```

```
        printLL(head2);

}
```

Output:



```
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques12.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
7
Enter the data
30
Enter the data
29
Enter the data
73
Enter the data
80
Enter the data
70
Enter the data
110
Enter the data
89
The source linked list is:
30->29->73->80->70->110->89
The destination linked list is:
30->29->70->110->89
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ |
```

Q.13. WAP that accepts a source singly linked list and a destination singly linked list and check whether the source list is a sublist of the destination list. Return the first position at which the sublist is found:

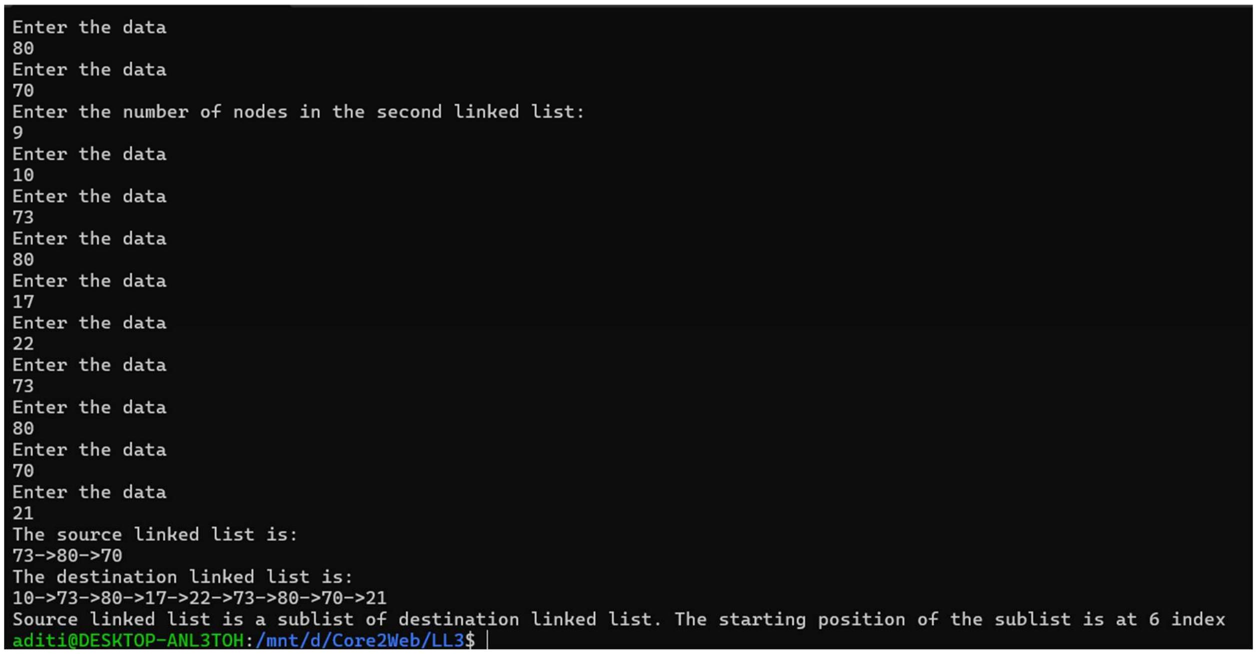
Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node *next;

};


int countNodes(struct node *head)

{
        int count=0;
        if(head==NULL)
        {
                return 0;
        }
        else
        {
                struct node *temp=head;
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                return count;
        }
}

int find_sublist(struct node *head1,struct node *head2)
{
        if(head2==NULL || head1==NULL)
        {
                printf("Either of the linked is empty therefore cannot find the sublist.\n");
                return -1;
        }
        int count_src=countNodes(head1);
        int count_des=countNodes(head2);
        if(count_src>count_des)
        {
                printf("Source linked list has more nodes than the destination linked list therefore, no
sublist possible\n");
                return -1;
        }
        else
        {
        struct node *ptr1=head1;
        struct node *ptr2=head2;
        int index=1;

        while(ptr2!=NULL)
        {
                struct node *temp1=ptr1;
                struct node *temp2=ptr2;
                while(temp1!=NULL && temp2!=NULL && temp1->data==temp2->data)
                {
                        temp1=temp1->next;
                        temp2=temp2->next;
                }
                if(temp1==NULL)
                {
                        return index;
                }
                index++;
                ptr2=ptr2->next;
        }
        return -1;
}
```

```
}

struct node *createNode()
{
	struct node *newNode=(struct node*)malloc(sizeof(struct node));
	printf("Enter the data\n");
	scanf("%d",&newNode->data);
	newNode->next=NULL;
	return newNode;
}

struct node *addNode(struct node *head)
{
	struct node *newNode=createNode();
	if(head==NULL)
	{
		head=newNode;
	}
	else
	{
		struct node *temp=head;
		while(temp->next!=NULL)
		{
			temp=temp->next;
		}
		temp->next=newNode;
	}
	return head;
}
```

```
void printLL(struct node *head)
{
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
}

void main()
{
        struct node *head1=NULL;
        struct node *head2=NULL;
        int c1,c2;
        printf("Enter the number of nodes in the first linked list:\n");
        scanf("%d",&c1);
        for(int i=1;i<=c1;i++)
        {
                head1=addNode(head1);
        }
        printf("Enter the number of nodes in the second linked list:\n");
        scanf("%d",&c2);
        for(int i=1;i<=c2;i++)
        {
                head2=addNode(head2);
        }
        printf("The source linked list is:\n");

        printLL(head1);

        printf("The destination linked list is:\n");

        printLL(head2);

        int pos=find_sublist(head1,head2);

        if(pos==-1)

        {

                printf("The source list is not a sublist of the destination list\n");


        }

        else

        {

                printf("Source linked list is a sublist of destination linked list. The starting position of the
sublist is at %d index\n",pos);

        }

}
```

Output:



```
Enter the data
80
Enter the data
70
Enter the number of nodes in the second linked list:
9
Enter the data
10
Enter the data
73
Enter the data
80
Enter the data
17
Enter the data
22
Enter the data
73
Enter the data
80
Enter the data
70
Enter the data
21
The source linked list is:
73->80->70
The destination linked list is:
10->73->80->17->22->73->80->70->21
Source linked list is a sublist of destination linked list. The starting position of the sublist is at 6 index
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$
```

Q.14. WAP that takes source singly linked list and destination singly linked list from the user and check whether source list is a sublist of the destination list and return the last index of the sublist in the destination list:

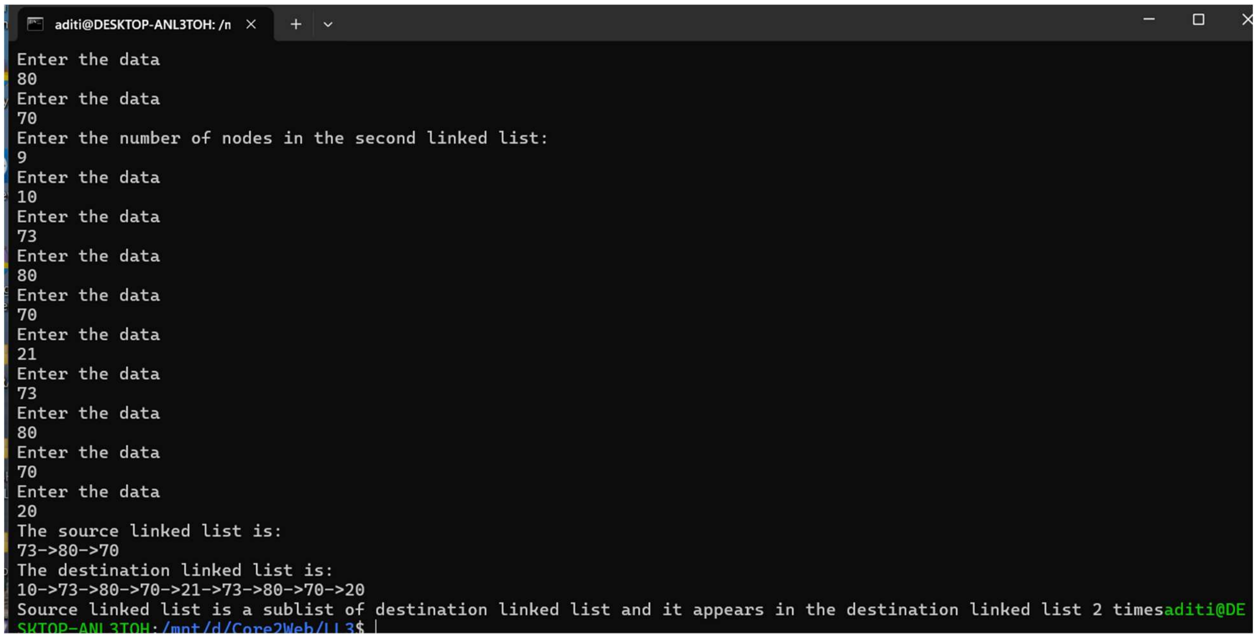
Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

	int data;

	struct node *next;

};


int countNodes(struct node *head)

{

	int count=0;

	if(head==NULL)

	{

		return 0;

	}

	else

	{

		struct node *temp=head;

		while(temp!=NULL)

		{

			count++;

			temp=temp->next;

		}

		return count;
		}
	}

	int find_sublist(struct node *head1,struct node *head2)
	{
		if(head2==NULL || head1==NULL)
		{
			printf("Either of the linked is empty therefore cannot find the sublist.\n");
			return -1;
		}
		int count_src=countNodes(head1);
		int count_des=countNodes(head2);
		if(count_src>count_des)
		{
			printf("Source linked list has more nodes than the destination linked list therefore, no
	sublist possible\n");
			return -1;
		}
		else
		{
		struct node *ptr1=head1;
		struct node *ptr2=head2;
		int index=1;
		int count=0;

		while(ptr2!=NULL)
		{
			struct node *temp1=ptr1;
			struct node *temp2=ptr2;
```

```
                while(temp1!=NULL && temp2!=NULL && temp1->data==temp2->data)
                {
                        temp1=temp1->next;
                        temp2=temp2->next;
                }
                if(temp1==NULL)
                {
                        count++;
                }

                ptr2=ptr2->next;
        }
        return count;
}

}

struct node *createNode()
{
        struct node *newNode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data\n");
        scanf("%d",&newNode->data);
        newNode->next=NULL;
        return newNode;
}

struct node *addNode(struct node *head)
{
        struct node *newNode=createNode();
        if(head==NULL)
        {
                head=newNode;
        }
        else
        {
                struct node *temp=head;
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=newNode;
        }
        return head;
}

void printLL(struct node *head)
{
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                printf("%d->",temp->data);
                temp=temp->next;
        }
        printf("%d\n",temp->data);
}

void main()
{
	struct node *head1=NULL;

	struct node *head2=NULL;

	int c1,c2;

	printf("Enter the number of nodes in the first linked list:\n");

	scanf("%d",&c1);

	for(int i=1;i<=c1;i++)

	{

		head1=addNode(head1);

	}

	printf("Enter the number of nodes in the second linked list:\n");

	scanf("%d",&c2);

	for(int i=1;i<=c2;i++)

	{

		head2=addNode(head2);

	}

	printf("The source linked list is:\n");

	printLL(head1);

	printf("The destination linked list is:\n");

	printLL(head2);

	int pos=find_sublist(head1,head2);

	if(pos==-1)

	{

		printf("The source list is not a sublist of the destination list\n");


	}

	else

	{

		printf("Source linked list is a sublist of destination linked list and it appears in the 
destination linked list %d times",pos);
```

```
        }
}
```

Output:



```
Enter the data
80
Enter the data
70
Enter the number of nodes in the second linked list:
9
Enter the data
10
Enter the data
73
Enter the data
80
Enter the data
70
Enter the data
21
Enter the data
73
Enter the data
80
Enter the data
70
Enter the data
20
The source linked list is:
73->80->70
The destination linked list is:
10->73->80->70->21->73->80->70->20
Source linked list is a sublist of destination linked list and it appears in the destination linked list 2 timesaditi@DE
SKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$
```

Q.15. Take a source singly linked list from the user and copy its contents in the destination linked list in ascending order:

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

        int data;

        struct node *next;

};


int countNode(struct node*);
```

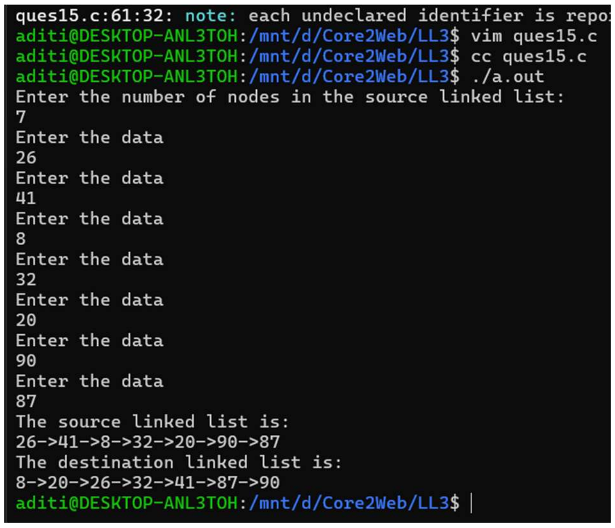
```
struct node *addNode(struct node*);

struct node *createNode();

void printLL(struct node*);

struct node *src_to_dest(struct node*, struct node*);



int countNode(struct node *head)

{

        if(head==NULL)

        {

                return 0;

        }

        else

        {

                struct node *temp=head;

                int count=0;

                while(temp!=NULL)

                {

                        count++;

                        temp=temp->next;

                }

                return count;

        }

}


struct node *src_to_dest(struct node *head1, struct node *head2)

{

        int count_src=countNode(head1);

        if(head1==NULL)
```

```
{
        printf("Linked List empty\n");
        return NULL;
}
else
{
        struct node *temp1=head1;
        while(temp1!=NULL)
        {
                struct node *newNode=(struct node*)malloc(sizeof(struct node));
                newNode->data=temp1->data;
                newNode->next=NULL;

                if(head2==NULL || head2->data>newNode->data)
                {
                        newNode->next=head2;
                        head2=newNode;
                }
                else
                {
                        struct node *temp2=head2;
                        while(temp2->next!=NULL && temp2->next->data<newNode->data)
                        {
                                temp2=temp2->next;
                        }
                        newNode->next=temp2->next;
                        temp2->next=newNode;
                }
                temp1=temp1->next;
```

```
                    }
                    return head2;
          }

}

struct node *createNode()
{
          struct node *newNode=(struct node*)malloc(sizeof(struct node));
          printf("Enter the data\n");
          scanf("%d",&newNode->data);
          newNode->next=NULL;
          return newNode;
}

struct node *addNode(struct node *head)
{
          struct node *newNode=createNode();
          if(head==NULL)
          {
                    head=newNode;
          }
          else
          {
                    struct node *temp=head;
                    while(temp->next!=NULL)
                    {
                              temp=temp->next;
                    }
                    temp->next=newNode;
          }
          return head;
}

void printLL(struct node *head)
{
          if(head==NULL)
          {
                    printf("Linked list empty\n");
          }
          else
          {
          struct node *temp=head;
          while(temp->next!=NULL)
          {
                    printf("%d->",temp->data);
                    temp=temp->next;
          }
          printf("%d\n",temp->data);
          }
}

void main()
{
          struct node *head1=NULL;
          struct node *head2=NULL;
          int c;
          printf("Enter the number of nodes in the source linked list:\n");
```

```
        scanf("%d",&c);
        for(int i=1;i<=c;i++)
        {
                head1=addNode(head1);
        }
        printf("The source linked list is:\n");
        printLL(head1);
        head2=src_to_dest(head1,head2);
        printf("The destination linked list is:\n");
        printLL(head2);

}
```

Output:



```
ques15.c:61:32: note: each undeclared identifier is repo
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ vim ques15.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ cc ques15.c
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$ ./a.out
Enter the number of nodes in the source linked list:
7
Enter the data
26
Enter the data
41
Enter the data
8
Enter the data
32
Enter the data
20
Enter the data
90
Enter the data
87
The source linked list is:
26->41->8->32->20->90->87
The destination linked list is:
8->20->26->32->41->87->90
aditi@DESKTOP-ANL3TOH:/mnt/d/Core2Web/LL3$
```