Linked List Assignment

question 1:

code :

```
//Question 1: Find no of occurance of entered element

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head=NULL;

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode() {

	node *new = createNode();

	if(head == NULL) {
		head = new;
	}else {

		node *tmp = head;
		while(tmp->next != NULL)
			tmp = tmp->next;
		tmp->next = new;
	}
}

void noOfOcc() {
	int num;
	printf("Enter element:");
	scanf("%d",&num);
	int cnt=0;
	node *tmp = head;

	while(tmp != NULL) {

		if(tmp->data == num)
			cnt++;

		tmp = tmp->next;
	}

	printf("\n%d is occured %d times\n",num,cnt);
}
```

```
void printList() {
	node *tmp = head;
	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

void main() {

	int ch;

	while(1) {

		printf("\n1.addNode\n");
		printf("2.noOfOccurance\n");
		printf("3.PrintList\n");
		printf("4.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode();
				break;
			case 2:
				noOfOcc();
				break;
			case 3:
				printList();
				break;
			case 4:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:10

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:20

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:30

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:40

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:30
```

```
1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:30

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:1
Enter number:30

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:2
Enter element:30

30 is occured 4 times

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:2
Enter element:100

100 is occured 0 times
```

```
1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:3
|10|->|20|->|30|->|40|->|30|->|30|->|30|->NULL

1.addNode
2.noOfOccurance
3.PrintList
4.Exit

Select option:4
```

Question 2:
Code:

```
//Question 2: concat two linked list

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

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

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;
        return new;
}

void addNode(node **head) {

        node *new = createNode();
        if(*head == NULL) {
                *head = new;
        }else {

                node *tmp = *head;
                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}


void concatList(node **dest , node **src) {

        node *tmp = *dest;

        if(*dest != NULL) {                     //if destination have nodes

                node *tmp = *dest;
                while(tmp->next != NULL)
```

```
                tmp = tmp->next;

            tmp->next = *src;

        }else {                 // if destination list empty , we will directly connect source list
            *dest = *src;
        }

        printf("After concatenation 2 lists are:\n");
        printf("=================================================================\n");
        printf("Destination List :\n");
        printList(*dest);
        printf("Source List :\n");
        printList(*src);
        printf("=================================================================\n");
}


void main() {

        // here 2nd list is destination 1nd 1st is source

        int ch;

        while(1) {

            printf("\n1.addNode in 1st list\n");
            printf("2.addNode in 2nd list\n");
            printf("3.Print 1st list\n");
            printf("4.Print 2nd list\n");
            printf("5.Concat two lists\n");
            printf("6.Exit\n");

            printf("\nSelect option:");
            scanf("%d",&ch);

            switch(ch) {

                case 1:
                    addNode(&head1);
                    break;
                case 2:
                    addNode(&head2);
                    break;
                case 3:
                    printList(head1);
                    break;
                case 4:
                    printList(head2);
                    break;
                case 5:
                    concatList(&head2,&head1);
                    break;
                case 6:
                    exit(0);
                    break;
            }

        }

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:10
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:20

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:40

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:3
|30|->|30|->|70|->NULL
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:4
|10|->|20|->|30|->|40|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:5
After concatenation 2 lists are:
================================================================
Destination List :
|10|->|20|->|30|->|40|->|30|->|30|->|70|->NULL
Source List :
|30|->|30|->|70|->NULL
================================================================

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:6
```

Question 3:
Code :

```
//Question 3: concat 1st n nodes of linked list

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

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

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;
	return new;
}

void addNode(node **head) {

	node *new = createNode();
	if(*head == NULL) {
		*head = new;
	}else {

		node *tmp = *head;
		while(tmp->next != NULL)
			tmp = tmp->next;
		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

node* copyNode(node *snode) {

	node *new = (node*)malloc(sizeof(node));
	new->data = snode->data;
	new->next = NULL;

	return new;
}
```

```
int countNodes(node *head) {

	int cnt=0;

	while(head != NULL) {

		cnt++;
		head = head->next;
	}

	return cnt;
}


void concatFirstNNodes(node **dest , node **src , int n) {

	node *tmp1 = *dest ,*tmp2 = *src;
	int cnt = countNodes(*src);			//count of nodes in 2nd list


	if(*dest != NULL) {			//if destination have nodes
		while(tmp1->next != NULL)
			tmp1 = tmp1->next;

	}

	if(*src != NULL) {

		if(n < cnt) {					//entered r2 is less or equal to present nodes

			while(n) {

				if(*dest == NULL){

					*dest = copyNode(tmp2);
					tmp1 = *dest;
					tmp2=tmp2->next;
				}else {
					tmp1->next = copyNode(tmp2);
					tmp1=tmp1->next;
					tmp2=tmp2->next;
				}

				n--;
			}

		}else {	//if range is not withing limit

			if(*dest == NULL)
				*dest = *src;
		}
	}

	printf("After concatenation 2 lists are:\n");
	printf("=============================================================\
n");
	printf("Destination List :\n");
	printList(*dest);
	printf("Source List :\n");
	printList(*src);
	printf("=============================================================\
n");
}
```

```
void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.addNode in 2nd list\n");
		printf("3.Print 1st list\n");
		printf("4.Print 2nd list\n");
		printf("5.Concat two lists\n");
		printf("6.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				addNode(&head2);
				break;
			case 3:
				printList(head1);
				break;
			case 4:
				printList(head2);
				break;
			case 5:{
				int n;
				printf("Enter no of nodes to concat:");
				scanf("%d",&n);
				concatFirstNNodes(&head2,&head1,n);
			      }break;
			case 6:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:10
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:20

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:40

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:3
|30|->|30|->|70|->NULL
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:4
|10|->|20|->|30|->|40|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:5
Enter no of nodes to concat:2
After concatenation 2 lists are:
=================================================================
Destination List :
|10|->|20|->|30|->|40|->|30|->|30|->NULL
Source List :
|30|->|30|->|70|->NULL
=================================================================

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:6
```

Question 4:

Code:

```
//Question 4: concat last n nodes of linked list

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

int countNodes(node *head) {

        int cnt=0;
```

```
		while(head != NULL) {

			cnt++;
			head = head->next;
		}

		return cnt;
	}


	void concatLastNNodes(node **dest , node **src , int n) {

		node *tmp1 = *dest ,*tmp2 = *src;
		int cnt = countNodes(*src);

		if(*src != NULL) {

			if(n < cnt) {

				while(cnt - n) {

					tmp2 = tmp2->next;
					cnt--;
				}
			}
		}else {

			printf("Src list is empty\n");
		}

		if(*dest != NULL) {
			while(tmp1->next != NULL)
				tmp1=tmp1->next;


				tmp1->next = tmp2;

		}else {				//if destination is empty

			*dest = tmp2;
		}

		printf("After concatenation 2 lists are:\n");
		printf("=================================================================\
n");
		printf("Destination List :\n");
		printList(*dest);
		printf("Source List :\n");
		printList(*src);
		printf("=================================================================\
n");
	}


	void main() {

		// here 2nd list is destination 1nd 1st is source

		int ch;

		while(1) {
```

```
                printf("\n1.addNode in 1st list\n");
                printf("2.addNode in 2nd list\n");
                printf("3.Print 1st list\n");
                printf("4.Print 2nd list\n");
                printf("5.Concat two lists\n");
                printf("6.Exit\n");

                printf("\nSelect option:");
                scanf("%d",&ch);

                switch(ch) {

                        case 1:
                                addNode(&head1);
                                break;
                        case 2:
                                addNode(&head2);
                                break;
                        case 3:
                                printList(head1);
                                break;
                        case 4:
                                printList(head2);
                                break;
                        case 5:{
                                int n;
                                printf("Enter n nodes:");
                                scanf("%d",&n);
                                concatLastNNodes(&head2,&head1,n);
                             }break;
                        case 6:
                                exit(0);
                                break;
                }

        }

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:10
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:20

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:40

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:3
|30|->|30|->|70|->NULL
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:4
|10|->|20|->|30|->|40|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:5
Enter n nodes:2
After concatenation 2 lists are:
================================================================
Destination List :
|10|->|20|->|30|->|40|->|30|->|70|->NULL
Source List :
|30|->|30|->|70|->NULL
================================================================

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:6
```

Question 5:
Code :

```
//Question 5: concat n nodes of linked list within a range

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

node* copyNode(node *snode) {

        node *new = (node*)malloc(sizeof(node));
        new->data = snode->data;
```

```
	new->next = NULL;

	return new;
}

int countNodes(node *head) {

	int cnt=0;

	while(head != NULL) {

		cnt++;
		head = head->next;
	}

	return cnt;
}


void concatRangeOfNNodes(node **dest , node **src , int r1,int r2) {

	node *tmp1 = *dest ,*tmp2 = *src;
	int cnt = countNodes(*src);			//count of nodes in 2nd list
	int tr1=r1;					//temporary var to store r1 for further operations

	if(r1 <= 0 || r2 > cnt || r1 > r2) {

		printf("Invaid range\n");
		return ;
	}

	if(*dest != NULL) {			//if destination have nodes
		while(tmp1->next != NULL)
			tmp1 = tmp1->next;

	}

	if(*src != NULL) {

		if(r2 <= cnt) {				//entered r2 is less or equal to present nodes

			while(r1-1) {
				tmp2 = tmp2->next;
				r1--;
			}

			while(r2-tr1+1) {

				if(*dest == NULL){

					*dest = copyNode(tmp2);
					tmp1 = *dest;
					tmp2=tmp2->next;
				}else {
					tmp1->next = copyNode(tmp2);
					tmp1=tmp1->next;
					tmp2=tmp2->next;
				}

				r2--;
			}
		}
	}
```

```
	printf("After concatenation 2 lists are:\n");
	printf("=================================================================\n");
	printf("Destination List :\n");
	printList(*dest);
	printf("Source List :\n");
	printList(*src);
	printf("=================================================================\n");
}


void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.addNode in 2nd list\n");
		printf("3.Print 1st list\n");
		printf("4.Print 2nd list\n");
		printf("5.Concat two lists\n");
		printf("6.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				addNode(&head2);
				break;
			case 3:
				printList(head1);
				break;
			case 4:
				printList(head2);
				break;
			case 5:{
				int r1,r2;
				printf("Enter range\n");
				printf("r1:");
				scanf("%d",&r1);
				printf("r2:");
				scanf("%d",&r2);
				concatRangeOfNNodes(&head2,&head1,r1,r2);
			    }break;
			case 6:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:80
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:90

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:1
Enter number:100

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:30

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:2
Enter number:40
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:3
|30|->|30|->|70|->|80|->|90|->|100|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:4
|30|->|40|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:5
Enter range
r1:2
r2:5
After concatenation 2 lists are:
================================================================
Destination List :
|30|->|40|->|30|->|70|->|80|->|90|->NULL
Source List :
|30|->|30|->|70|->|80|->|90|->|100|->NULL
================================================================

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Concat two lists
6.Exit

Select option:6
```

Question 6:

Code:

```
//Question 6: copy nodes of src to dest linked list

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode(node **head) {

	node *new = createNode();

	if(*head == NULL) {

		*head = new;
	}else {

		node *tmp = *head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

node* copyNode(node *snode) {
```

```
        node *new = (node*)malloc(sizeof(node));
        new->data = snode->data;
        new->next = NULL;

        return new;
}

int countNodes(node *head) {

        int cnt=0;

        while(head != NULL) {

                cnt++;
                head = head->next;
        }

        return cnt;
}


void copyFirstNNodes(node **dest , node **src , int n) {

        node *tmp1 = *dest,*tmp2=*src;
        int cnt = countNodes(*src);                     //count of nodes in src list

         if(*dest != NULL) {                    //if destination have nodes this is helful when we are copying 2nd
time
       while(tmp1->next != NULL)
           tmp1 = tmp1->next;

 }

        if(*src != NULL) {

                if(n < cnt) {                                   //entered n is less than present nodes

                        while(n) {

                                if(*dest == NULL){

                                        *dest = copyNode(tmp2);
                                        tmp1 = *dest;
                                        tmp2=tmp2->next;
                                }else {
                                        tmp1->next = copyNode(tmp2);
                                        tmp1=tmp1->next;
                                        tmp2=tmp2->next;
                                }

                                n--;
                        }

                }else {   //if n > cnt

                                *dest = *src;
                }
        }

        printf("After copying 2 lists are:\n");
        printf("================================================================\
n");
```

```
		printf("Destination List :\n");
		printList(*dest);
		printf("Source List :\n");
		printList(*src);
		printf("================================================================\
n");
	}


	void main() {

		// here 2nd list is destination 1nd 1st is source

		int ch;

		while(1) {

			printf("\n1.addNode in 1st list\n");
			printf("2.Print 1st list\n");
			printf("3.Print 2nd list\n");
			printf("4.copy first n nodes \n");
			printf("5.Exit\n");

			printf("\nSelect option:");
			scanf("%d",&ch);

			switch(ch) {

				case 1:
					addNode(&head1);
					break;
				case 2:
					printList(head1);
					break;
				case 3:
					printList(head2);
					break;
				case 4:{
					int n;
					printf("Enter no of nodes to copy:");
					scanf("%d",&n);
					copyFirstNNodes(&head2,&head1,n);
				  }break;
				case 5:
					exit(0);
					break;
			}

		}

	}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit
```

```
Select option:1
Enter number:90

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:1
Enter number:100

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:2
|30|->|30|->|70|->|80|->|90|->|100|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:3
NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit
```

```
Select option:4
Enter no of nodes to copy:4
After copying 2 lists are:
=====================================================================
Destination List :
|30|->|30|->|70|->|80|->NULL
Source List :
|30|->|30|->|70|->|80|->|90|->|100|->NULL
=====================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy first n nodes
5.Exit

Select option:5
```

Question 7:

Code:

```
//Question 7: copy last n nodes of linked list

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode(node **head) {

	node *new = createNode();

	if(*head == NULL) {

		*head = new;
	}else {

		node *tmp = *head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

int countNodes(node *head) {
```

```
	int cnt=0;

	while(head != NULL) {

		cnt++;
		head = head->next;
	}

	return cnt;
}


void copyLastNNodes(node **dest , node **src , int n) {

	node *tmp1 = *dest ,*tmp2 = *src;
	int cnt = countNodes(*src);

	if(*src != NULL) {

		if(n < cnt) {

			while(cnt - n) {

				tmp2 = tmp2->next;
				cnt--;
			}
		}
	}else {

		printf("Src list is empty\n");
	}

	if(*dest != NULL) {					//this is helpful when we copy 2nd time
		while(tmp1->next != NULL)
			tmp1=tmp1->next;
			tmp1->next = tmp2;

	}else {				//if destination is empty

		*dest = tmp2;
	}

	printf("After copying 2 lists are:\n");
	printf("=================================================================\
n");
	printf("Destination List :\n");
	printList(*dest);
	printf("Source List :\n");
	printList(*src);
	printf("=================================================================\
n");
}


void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
```

```
                printf("2.Print 1st list\n");
                printf("3.Print 2nd list\n");
                printf("4.copy last n nodes\n");
                printf("5.Exit\n");

                printf("\nSelect option:");
                scanf("%d",&ch);

                switch(ch) {

                        case 1:
                                addNode(&head1);
                                break;
                        case 2:
                                printList(head1);
                                break;
                        case 3:
                                printList(head2);
                                break;
                        case 4:{
                                int n;
                                printf("Enter n nodes:");
                                scanf("%d",&n);
                                copyLastNNodes(&head2,&head1,n);
                             }break;
                        case 5:
                                exit(0);
                                break;
                }

        }

}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:90
```

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:1
Enter number:100

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:2
|30|->|30|->|70|->|80|->|90|->|100|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:3
NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:4
Enter n nodes:4
After copying 2 lists are:
================================================================
Destination List :
|70|->|80|->|90|->|100|->NULL
Source List :
|30|->|30|->|70|->|80|->|90|->|100|->NULL
================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy last n nodes
5.Exit

Select option:5
```

Question 8:
Code:

```
//Question 8: copy n nodes of linked list within a range

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

node* copyNode(node *snode) {

        node *new = (node*)malloc(sizeof(node));
```

```
	new->data = snode->data;
	new->next = NULL;

	return new;
}

int countNodes(node *head) {

	int cnt=0;

	while(head != NULL) {

		cnt++;
		head = head->next;
	}

	return cnt;
}


void copyRangeOfNNodes(node **dest , node **src , int r1,int r2) {

	node *tmp1 = *dest ,*tmp2 = *src;
	int cnt = countNodes(*src);			//count of nodes in 2nd list
	int tr1=r1;					//temporary var to store r1 for further operations

	if(r1 <= 0 || r2 > cnt || r1 > r2) {

		printf("Invaid range\n");
		return ;
	}

	if(*dest != NULL) {			//if destination have nodes helful for copying multiple times
		while(tmp1->next != NULL)
			tmp1 = tmp1->next;

	}

	if(*src != NULL) {

		if(r2 <= cnt) {				//entered r2 is less or equal to present nodes

			while(r1-1) {
				tmp2 = tmp2->next;
				r1--;
			}

			while(r2-tr1+1) {

				if(*dest == NULL){

					*dest = copyNode(tmp2);
					tmp1 = *dest;
					tmp2=tmp2->next;
				}else {
					tmp1->next = copyNode(tmp2);
					tmp1=tmp1->next;
					tmp2=tmp2->next;
				}

				r2--;
			}
```

```
		}

		printf("After copying 2 lists are:\n");
		printf("=================================================================\
n");
		printf("Destination List :\n");
		printList(*dest);
		printf("Source List :\n");
		printList(*src);
		printf("=================================================================\
n");
	}

	}


	void main() {

		// here 2nd list is destination 1nd 1st is source

		int ch;

		while(1) {

			printf("\n1.addNode in 1st list\n");
			printf("2.Print 1st list\n");
			printf("3.Print 2nd list\n");
			printf("4.copy n nodes between range\n");
			printf("5.Exit\n");

			printf("\nSelect option:");
			scanf("%d",&ch);

			switch(ch) {

				case 1:
					addNode(&head1);
					break;
				case 2:
					printList(head1);
					break;
				case 3:
					printList(head2);
					break;
				case 4:{
					int r1,r2;
					printf("Enter range\n");
					printf("r1:");
					scanf("%d",&r1);
					printf("r2:");
					scanf("%d",&r2);
					copyRangeOfNNodes(&head2,&head1,r1,r2);
				  }break;
				case 5:
					exit(0);
					break;
			}

		}
	}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:1
Enter number:90

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit
```

```
Select option:1
Enter number:100

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:2
|30|->|30|->|70|->|80|->|90|->|100|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:3
NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:4
Enter range
r1:2
r2:5
After copying 2 lists are:
================================================================
Destination List :
|30|->|70|->|80|->|90|->NULL
Source List :
|30|->|30|->|70|->|80|->|90|->|100|->NULL
================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.copy n nodes between range
5.Exit

Select option:5
```

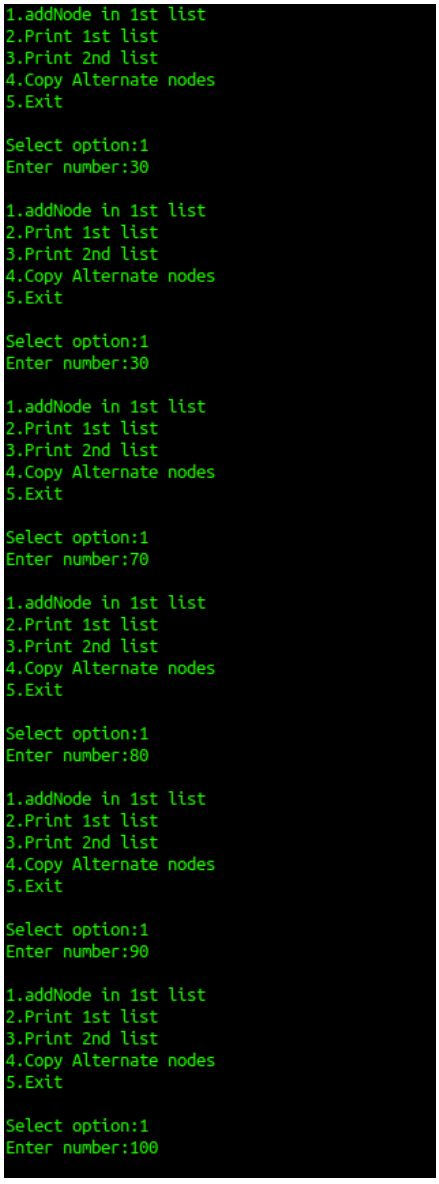
Question 9:
Code:

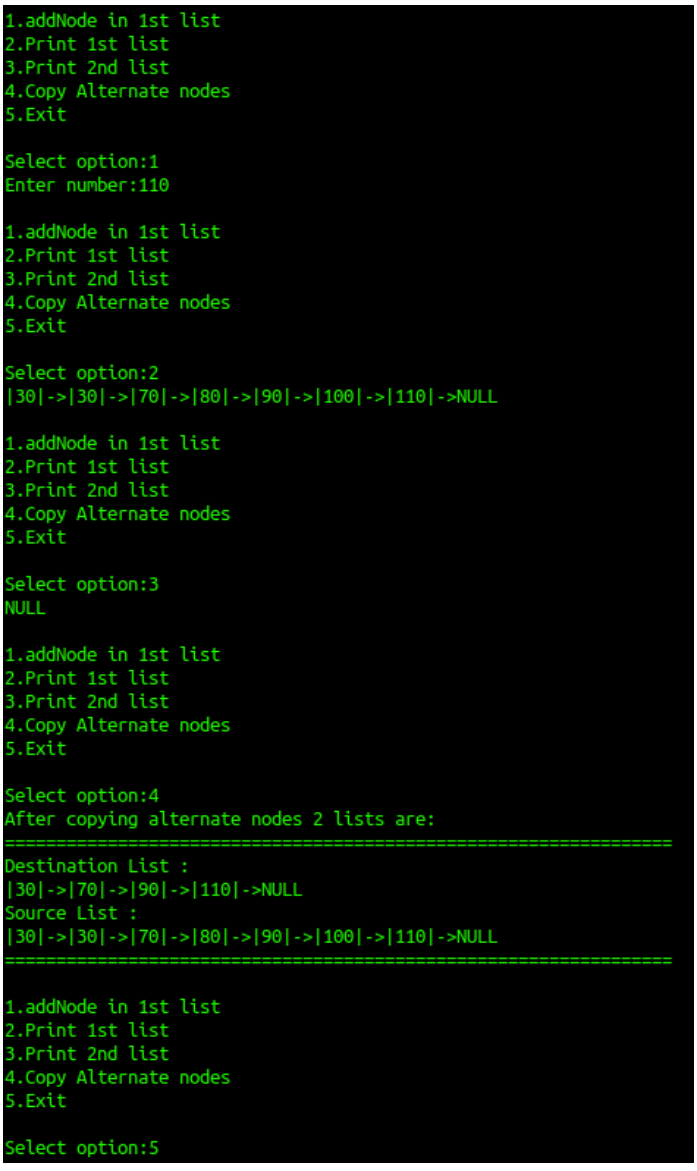
```
//Question 9: copy alternate nodes of linked list

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

node* copyNode(node *snode) {

  node *new = (node*)malloc(sizeof(node));
```

```
    new->data = snode->data;
    new->next = NULL;

    return new;
}

    void copyAlternateNodes(node **dest , node **src) {

            node *tmp1 = *dest , *tmp2 = *src;

            if(*src != NULL) {                  //if source have nodes

                    while(tmp2 != NULL && tmp2->next != NULL) {

                            if(*dest == NULL) {
                                    *dest = copyNode(tmp2);
                                    tmp1 = *dest;
                                    tmp2 = tmp2->next->next;
                            }else {

                                    tmp1->next = copyNode(tmp2);
                                    tmp1 = tmp1->next;
                                    tmp2 = tmp2->next->next;
                            }
                    }

                    tmp1->next = tmp2;


            }else {                         // if src list is empty
                    printf("Source linked list is empty\n");
            }

            printf("After copying alternate nodes 2 lists are:\n");
    printf("================================================================\n");
    printf("Destination List :\n");
    printList(*dest);
    printf("Source List :\n");
    printList(*src);
    printf("================================================================\n");

    }


    void main() {

            // here 2nd list is destination 1nd 1st is source

            int ch;

            while(1) {

                    printf("\n1.addNode in 1st list\n");
                    printf("2.Print 1st list\n");
                    printf("3.Print 2nd list\n");
                    printf("4.Copy Alternate nodes\n");
                    printf("5.Exit\n");

                    printf("\nSelect option:");
                    scanf("%d",&ch);

                    switch(ch) {
```

```
                    case 1:
                        addNode(&head1);
                        break;
                    case 2:
                        printList(head1);
                        break;
                    case 3:
                        printList(head2);
                        break;
                    case 4:
                        copyAlternateNodes(&head2,&head1);
                        break;
                    case 5:
                        exit(0);
                        break;
                }

            }

        }
```

Output:





Question 10:
Code:

```
//Question 10: Copy nodes data whose sum is even

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

node* copyNode(node *snode) {

  node *new = (node*)malloc(sizeof(node));
```

```
    new->data = snode->data;
    new->next = NULL;

    return new;
}

    int isEvenDigit(int num) {

            int sum=0;

            while(num != 0 ) {

                    sum = sum + num%10;
                    num = num/10;
            }

            if(sum%2 == 0)
                    return 1;
            else
                    return 0;
    }

    void copyNodes(node **dest , node **src) {

            node *tmp1 = *dest , *tmp2 = *src;

            if(*src != NULL) {                  //if source have nodes

                    while(tmp2 != NULL) {

                            if(isEvenDigit(tmp2->data)) {

                                    if(*dest == NULL) {
                                            *dest = copyNode(tmp2);
                                            tmp1 = *dest;
                                    }else {

                                            tmp1->next = copyNode(tmp2);
                                            tmp1 = tmp1->next;
                                    }
                            }

                            tmp2 = tmp2->next;
                    }

                    if(*dest != NULL)
                            tmp1->next = tmp2;


            }else {                         // if src list is empty
                    printf("Source linked list is empty\n");
            }

            printf("After copying nodes 2 lists are:\n");
     printf("================================================================\n");
     printf("Destination List :\n");
     printList(*dest);
     printf("Source List :\n");
     printList(*src);
     printf("================================================================\n");

    }
```

```
void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.Print 1st list\n");
		printf("3.Print 2nd list\n");
		printf("4.Copy nodes data whose sum is even\n");
		printf("5.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				printList(head1);
				break;
			case 3:
				printList(head2);
				break;
			case 4:
				copyNodes(&head2,&head1);
				break;
			case 5:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:33

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:73

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit
```

```
Select option:1
Enter number:90

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:100

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:1
Enter number:110

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:2
|30|->|33|->|73|->|80|->|90|->|100|->|110|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:3
NULL
```

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:4
After copying nodes 2 lists are:
================================================================
Destination List :
|33|->|73|->|80|->|110|->NULL
Source List :
|30|->|33|->|73|->|80|->|90|->|100|->|110|->NULL
================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose sum is even
5.Exit

Select option:5
```

Question 11:
Code:

```
//Question 11: Copy nodes data which is prime


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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}

node* copyNode(node *snode) {

  node *new = (node*)malloc(sizeof(node));
```

```
    new->data = snode->data;
    new->next = NULL;

    return new;
}

    int isPrime(int num) {

            int cnt=0;

            for(int i=2;i<=num;i++) {

                    if(num%i==0)
                            cnt++;
            }

            if(cnt <= 1)
                    return 1;
            else
                    return 0;
    }

    void copyPrimeNodes(node **dest , node **src) {

            node *tmp1 = *dest , *tmp2 = *src;

            if(*src != NULL) {                      //if source have nodes

                    while(tmp2 != NULL) {

                            if(isPrime(tmp2->data)) {

                                    if(*dest == NULL) {
                                            *dest = copyNode(tmp2);
                                            tmp1 = *dest;
                                    }else {

                                            tmp1->next = copyNode(tmp2);
                                            tmp1 = tmp1->next;
                                    }
                            }

                            tmp2 = tmp2->next;
                    }

                    if(*dest != NULL)
                            tmp1->next = tmp2;


            }else {                         // if src list is empty
                    printf("Source linked list is empty\n");
            }

            printf("After copying nodes 2 lists are:\n");
    printf("================================================================\n");
    printf("Destination List :\n");
    printList(*dest);
    printf("Source List :\n");
    printList(*src);
    printf("================================================================\n");

    }
```

```
void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.Print 1st list\n");
		printf("3.Print 2nd list\n");
		printf("4.Copy nodes data which is prime\n");
		printf("5.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				printList(head1);
				break;
			case 3:
				printList(head2);
				break;
			case 4:
				copyPrimeNodes(&head2,&head1);
				break;
			case 5:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:29

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:73

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit
```

```
Select option:1
Enter number:70

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:110

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:1
Enter number:89

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:2
|30|->|29|->|73|->|80|->|70|->|110|->|89|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:3
NULL
```

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:4
After copying nodes 2 lists are:
================================================================
Destination List :
|29|->|73|->|89|->NULL
Source List :
|30|->|29|->|73|->|80|->|70|->|110|->|89|->NULL
================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data which is prime
5.Exit

Select option:5
```

Question 12:

Code:

```
//Question 12: Copy nodes data whose digits sum is prime

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode(node **head) {

	node *new = createNode();

	if(*head == NULL) {

		*head = new;
	}else {

		node *tmp = *head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

node* copyNode(node *snode) {
```

```
    node *new = (node*)malloc(sizeof(node));
    new->data = snode->data;
    new->next = NULL;

    return new;
}

    int isPrime(int num) {

            int sum = 0;
            while(num != 0) {

                    sum = sum + num%10;
                    num = num / 10;
            }

            int cnt=0;

            for(int i=2;i<=sum;i++) {

                    if(sum%i==0)
                            cnt++;
            }

            if(cnt <= 1)
                    return 1;
            else
                    return 0;
    }

    void copyPrimeDigitsNodes(node **dest , node **src) {

            node *tmp1 = *dest , *tmp2 = *src;

            if(*src != NULL) {                      //if source have nodes

                    while(tmp2 != NULL) {

                            if(isPrime(tmp2->data)) {

                                    if(*dest == NULL) {
                                            *dest = copyNode(tmp2);
                                            tmp1 = *dest;
                                    }else {

                                            tmp1->next = copyNode(tmp2);
                                            tmp1 = tmp1->next;
                                    }
                            }

                            tmp2 = tmp2->next;
                    }

                    if(*dest != NULL)
                            tmp1->next = tmp2;


            }else {                         // if src list is empty
                    printf("Source linked list is empty\n");
            }

            printf("After copying nodes 2 lists are:\n");
    printf("================================================================\n");
```

```
 printf("Destination List :\n");
 printList(*dest);
 printf("Source List :\n");
 printList(*src);
 printf("================================================================\n");

}


void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.Print 1st list\n");
		printf("3.Print 2nd list\n");
		printf("4.Copy nodes data whose digits sum is prime\n");
		printf("5.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				printList(head1);
				break;
			case 3:
				printList(head2);
				break;
			case 4:
				copyPrimeDigitsNodes(&head2,&head1);
				break;
			case 5:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:30

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:29

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:73

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit
```

```
Select option:1
Enter number:70

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:110

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:1
Enter number:89

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:2
|30|->|29|->|73|->|80|->|70|->|110|->|89|->NULL

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:3
NULL
```

```
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:4
After copying nodes 2 lists are:
================================================================
Destination List :
|30|->|29|->|70|->|110|->|89|->NULL
Source List :
|30|->|29|->|73|->|80|->|70|->|110|->|89|->NULL
================================================================

1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy nodes data whose digits sum is prime
5.Exit

Select option:5
```

Question 13:

Code :

```
//Question 13: find sub-list and return 1st position of sub-list

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

typedef struct node {

        int data;
        struct node *next;
}node;

node *head1=NULL;       //for 1st linked list
node *head2=NULL;       //for 2nd linked list

node *createNode() {

        node *new = (node*)malloc(sizeof(node));
        printf("Enter number:");
        scanf("%d",&new->data);
        new->next=NULL;

        return new;
}

void addNode(node **head) {

        node *new = createNode();

        if(*head == NULL) {

                *head = new;
        }else {

                node *tmp = *head;

                while(tmp->next != NULL)
                        tmp = tmp->next;

                tmp->next = new;
        }
}

void printList(node *head) {

        node *tmp = head;

        while(tmp != NULL) {

                printf("|%d|->",tmp->data);
                tmp = tmp->next;
        }

        printf("NULL\n");
}


int countNodes(node *head) {

        int cnt=0;
```

```
		while(head != NULL) {

			cnt++;
			head = head->next;
		}

		return cnt;
	}


	int findSubList1(node **dest , node **src) {

		node *tmp1 = *dest ,*tmp2 = *src;
		int pos = -1,ptr=0;

		if(*src == NULL || *dest == NULL) {
			return 0;			//returning 0 cause -1 means sub-list not found and our list indexing is starting from 1
		}


			//if destination and source have nodes

			while(tmp2 != NULL && tmp1 != NULL ) {

				ptr++;

				if(tmp2->data == tmp1->data) {

					if(pos == -1)
						pos = ptr;

					tmp2 = tmp2->next;
				}else {					//if we got few first nodes and then got unmatched nodes then reset "pos" and "tmp2"

					if(pos != -1) {			//we can write below 2 statements directly but ,condition avoides overriding of values multiple times
						pos = -1;
						tmp2 = *src;
					}

				}
				tmp1 = tmp1->next;
			}

			if(tmp2 != NULL && pos != -1)			//we havent traverse whole src and still we get pos
				return -1;


		return pos;
	}


	void main() {

		// here 2nd list is destination 1nd 1st is source

		int ch;

		while(1) {
```

```
		printf("\n1.addNode in 1st list\n");
		printf("2.addNode in 2nd list\n");
		printf("3.Print 1st list\n");
		printf("4.Print 2nd list\n");
		printf("5.Find sub list's first position\n");
		printf("6.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				addNode(&head2);
				break;
			case 3:
				printList(head1);
				break;
			case 4:
				printList(head2);
				break;
			case 5:{
				int ret = findSubList1(&head2,&head1);

				if(ret == -1)
					printf("Sub list not fount\n");
				else if(ret == 0)
					printf("Dest/src is empty\n");
				else
					printf("First Sub list fount at %d position\n",ret);

			    }break;
			case 6:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:1
Enter number:73

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:10

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:73
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:80

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:17

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:22

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:73

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:80
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:2
Enter number:21

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:3
|73|->|80|->|70|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:4
|10|->|73|->|80|->|17|->|22|->|73|->|80|->|70|->|21|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's first position
6.Exit

Select option:5
First Sub list fount at 6 position
```

Question 14:

Code:

```
//Question 14: find sub-list and return last position of sub-list

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode(node **head) {

	node *new = createNode();

	if(*head == NULL) {

		*head = new;
	}else {

		node *tmp = *head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

int countNodes(node *head) {
```

```
	int cnt=0;

	while(head != NULL) {

		cnt++;
		head = head->next;
	}

	return cnt;
}


int findSubList2(node **dest , node **src) {

	node *tmp1 = *dest ,*tmp2 = *src;
	int pos = -1,ptr=0,tpos = -1;

	if(*src == NULL || *dest == NULL) {
		return 0;			//returning 0 cause -1 means list not found and our list indexing is
starting from 1
	}


		//if destination and source have nodes

		while(tmp1 != NULL ) {

			ptr++;

			if(tmp2->data == tmp1->data) {
				pos = ptr;
				tmp2 = tmp2->next;
			}else {					//if we got few first nodes and then got
unmatched nodes then reset "pos" and "tmp2"


				if(pos != -1) {			//we can write below 2 statements
directly but ,condition avoides overriding of values multiple times
					pos = -1;
					tmp2 = *src;
				}

			}

			if(tmp2 == NULL) {		//if src list is traverse completly
				tmp2 = *src;
				tpos = pos;
				pos = -1;
			}

			tmp1 = tmp1->next;
		}

		if(pos == -1 && tpos != -1)
			return tpos;


	return pos;
}


void main() {
```

```
	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.addNode in 2nd list\n");
		printf("3.Print 1st list\n");
		printf("4.Print 2nd list\n");
		printf("5.Find sub list's last position\n");
		printf("6.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				addNode(&head2);
				break;
			case 3:
				printList(head1);
				break;
			case 4:
				printList(head2);
				break;
			case 5:{
				int ret = findSubList2(&head2,&head1);

				if(ret == -1)
					printf("Sub list not fount\n");
				else if(ret == 0)
					printf("Dest/src is empty\n");
				else
					printf("Last Sub list fount at %d position\n",ret);

			      }break;
			case 6:
				exit(0);
				break;
		}

	}

}
```

Output:

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:1
Enter number:73

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:1
Enter number:80

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:1
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:10
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:73

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:80

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:22
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:73

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:80

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:70

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:2
Enter number:21
```

```
1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:3
|73|->|80|->|70|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:4
|10|->|73|->|80|->|70|->|22|->|73|->|80|->|70|->|21|->NULL

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:5
Last Sub list fount at 8 position

1.addNode in 1st list
2.addNode in 2nd list
3.Print 1st list
4.Print 2nd list
5.Find sub list's last position
6.Exit

Select option:6
```

Question 15:

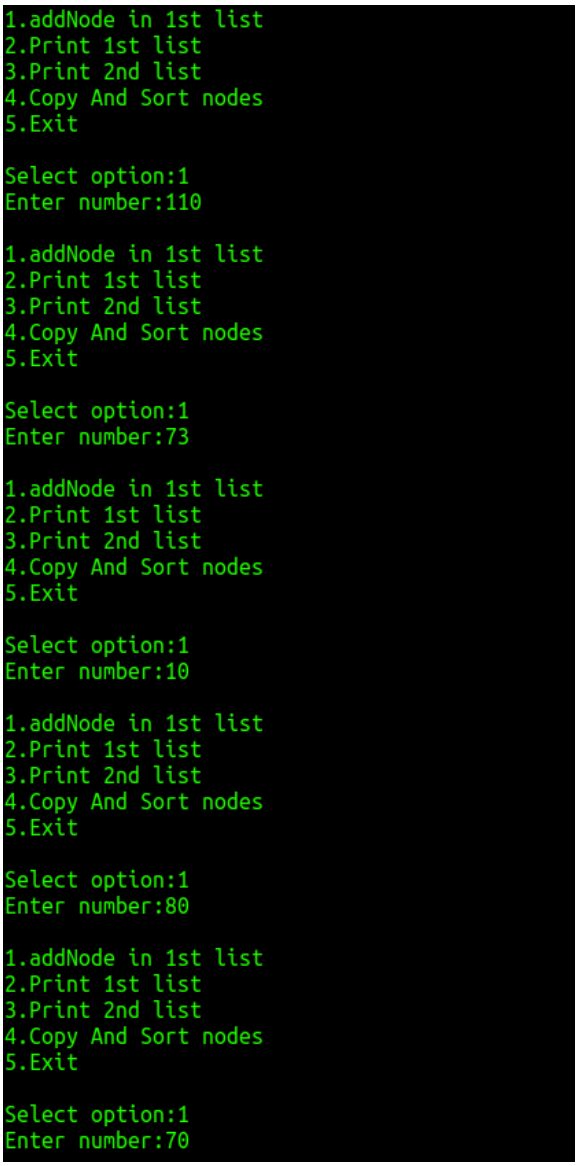
Code:

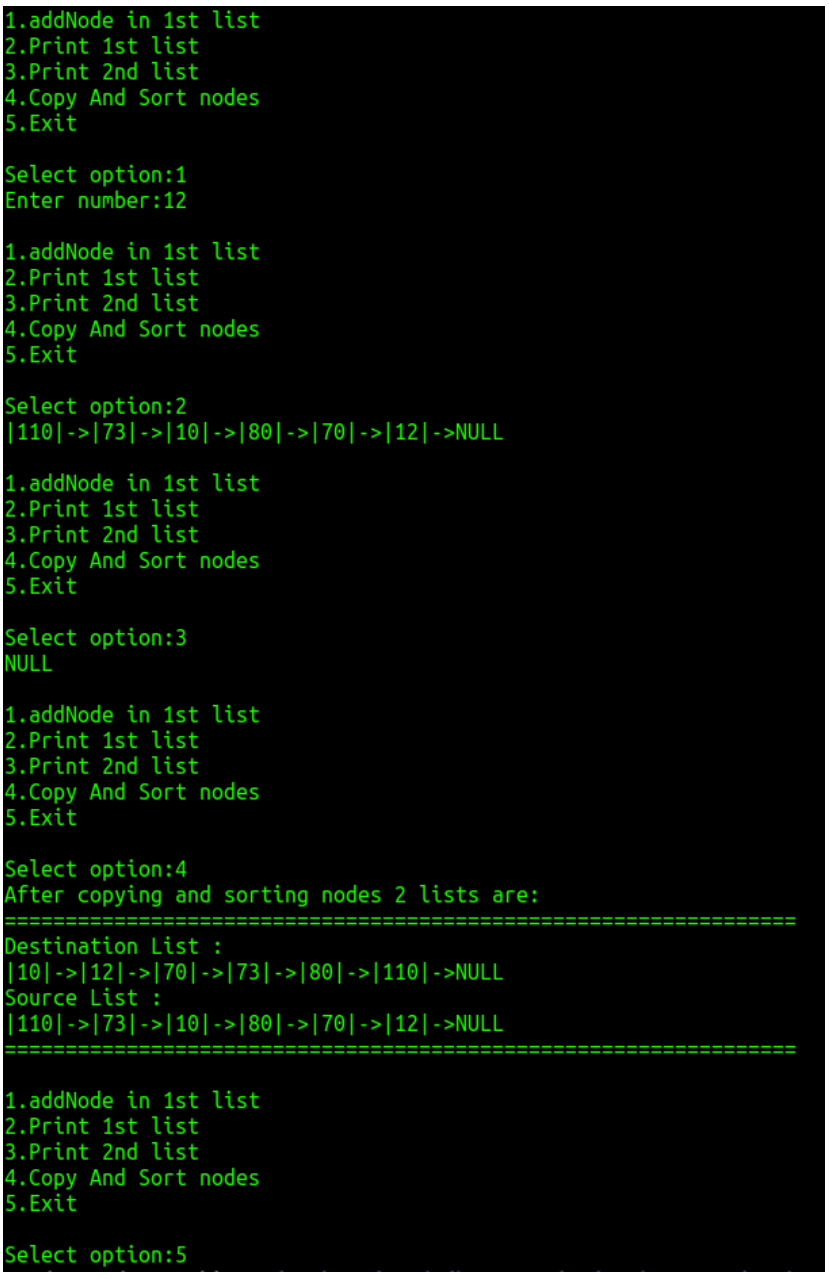
```
//Question 15: copy source list in destination and sort it

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

typedef struct node {

	int data;
	struct node *next;
}node;

node *head1=NULL;	//for 1st linked list
node *head2=NULL;	//for 2nd linked list

node *createNode() {

	node *new = (node*)malloc(sizeof(node));
	printf("Enter number:");
	scanf("%d",&new->data);
	new->next=NULL;

	return new;
}

void addNode(node **head) {

	node *new = createNode();

	if(*head == NULL) {

		*head = new;
	}else {

		node *tmp = *head;

		while(tmp->next != NULL)
			tmp = tmp->next;

		tmp->next = new;
	}
}

void printList(node *head) {

	node *tmp = head;

	while(tmp != NULL) {

		printf("|%d|->",tmp->data);
		tmp = tmp->next;
	}

	printf("NULL\n");
}

node* copyNode(node *snode) {
```

```
        node *new = (node*)malloc(sizeof(node));
        new->data = snode->data;
        new->next = NULL;

        return new;
    }

        void sortList(node **head) {

                node *tmp1 = *head ,*tmp2 = NULL;

                while(tmp1 != NULL) {

                        tmp2 = tmp1->next;

                        while( tmp2 != NULL) {

                                if(tmp1->data > tmp2->data) {

                                        int data = tmp1->data;
                                        tmp1->data = tmp2->data;
                                        tmp2->data = data;
                                }
                                tmp2 = tmp2->next;
                        }

                        tmp1 = tmp1->next;
                }
        }

        void copyAndSortNodes(node **dest , node **src) {

                if(*dest != NULL)                       //if we are calling copyAndSortNodes() multiple times dest should
be NULL everytime
                        *dest = NULL;

                node *tmp1 = *dest , *tmp2 = *src;

                if(*src != NULL) {                      //if source have nodes

                        while(tmp2 != NULL ) {

                                if(*dest == NULL) {
                                        *dest = copyNode(tmp2);
                                        tmp1 = *dest;
                                        tmp2 = tmp2->next;
                                }else {

                                        tmp1->next = copyNode(tmp2);
                                        tmp1 = tmp1->next;
                                        tmp2 = tmp2->next;
                                }
                        }

                        sortList(dest);

                }else {                                 // if src list is empty
                        printf("Source linked list is empty\n");
                }

                printf("After copying and sorting nodes 2 lists are:\n");
        printf("=================================================================\n");
        printf("Destination List :\n");
```

```
 printList(*dest);
 printf("Source List :\n");
 printList(*src);
 printf("=================================================================\n");

}


void main() {

	// here 2nd list is destination 1nd 1st is source

	int ch;

	while(1) {

		printf("\n1.addNode in 1st list\n");
		printf("2.Print 1st list\n");
		printf("3.Print 2nd list\n");
		printf("4.Copy And Sort nodes\n");
		printf("5.Exit\n");

		printf("\nSelect option:");
		scanf("%d",&ch);

		switch(ch) {

			case 1:
				addNode(&head1);
				break;
			case 2:
				printList(head1);
				break;
			case 3:
				printList(head2);
				break;
			case 4:
				copyAndSortNodes(&head2,&head1);
				break;
			case 5:
				exit(0);
				break;
		}

	}

}
```

Output:


1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:110
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:73
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:10
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:80
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:70


1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:1
Enter number:12
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:2
|110|->|73|->|10|->|80|->|70|->|12|->NULL
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:3
NULL
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:4
After copying and sorting nodes 2 lists are:
Destination List :
|10|->|12|->|70|->|73|->|80|->|110|->NULL
Source List :
|110|->|73|->|10|->|80|->|70|->|12|->NULL
1.addNode in 1st list
2.Print 1st list
3.Print 2nd list
4.Copy And Sort nodes
5.Exit
Select option:5