Ramakrishna Mission Vidyamandira

(An Autonomous College Under University of Calcutta)

Computer Science (Honors) Semester II 2024

Paper: 2CMSACOC1 Practical

Submitted by
Class Roll Number: 340 Registration Number: B.Sc. 2th Semester Batch: 2023-27

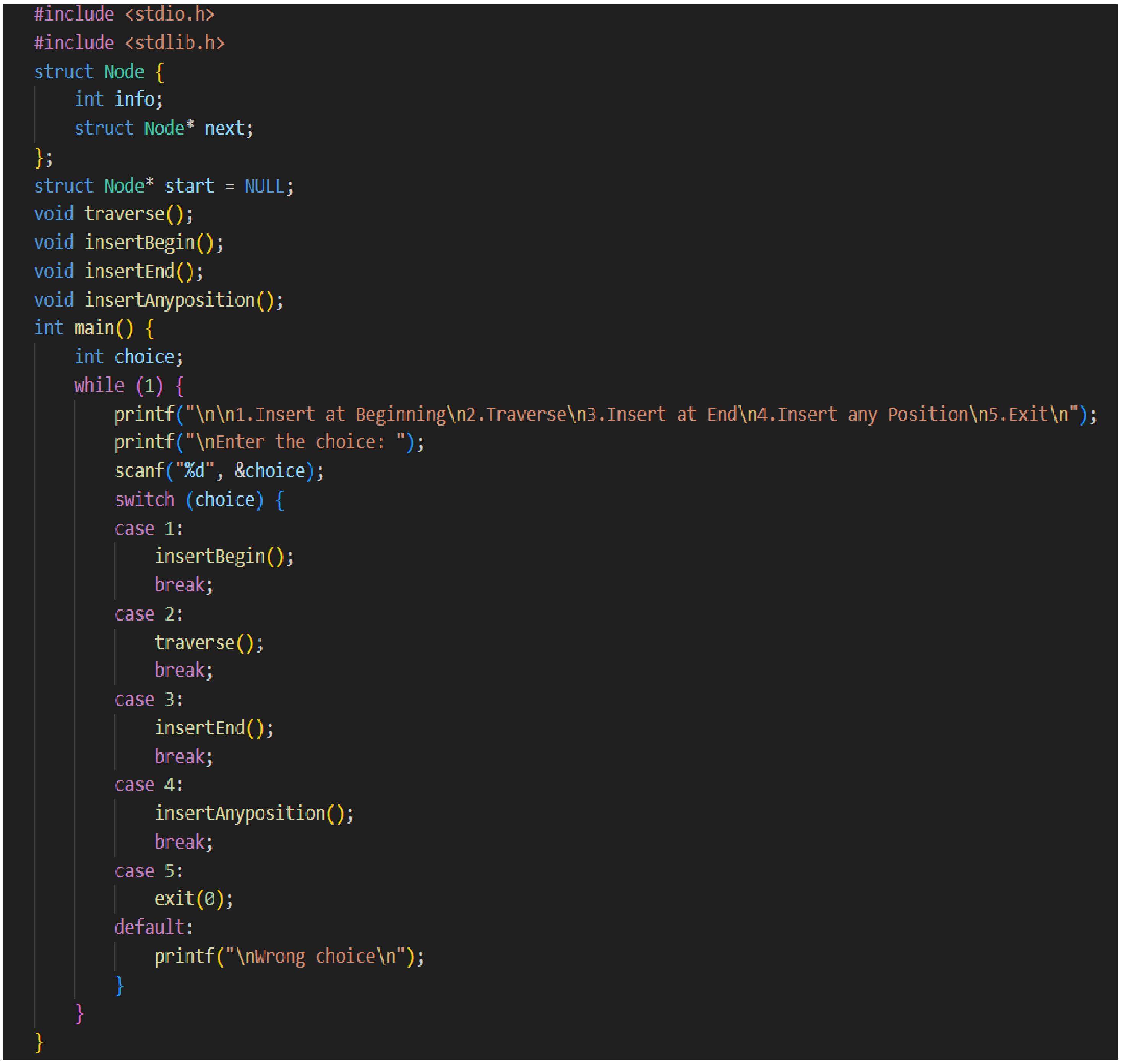
INDEX

SI NO.	ASSIGNMNET STATEMENT	D-O-A	D-O-S	SIGNATURE
1.	**Write a program to perform all insertion technique in a linear linked list.**			
2.	**Write a program to perform all insertion technique in a circular linked list.**			
3.	**Write a program to find odd number sum in a linked list.**			
4.	**Write a program to perform polynomial addition of two linked list.**			

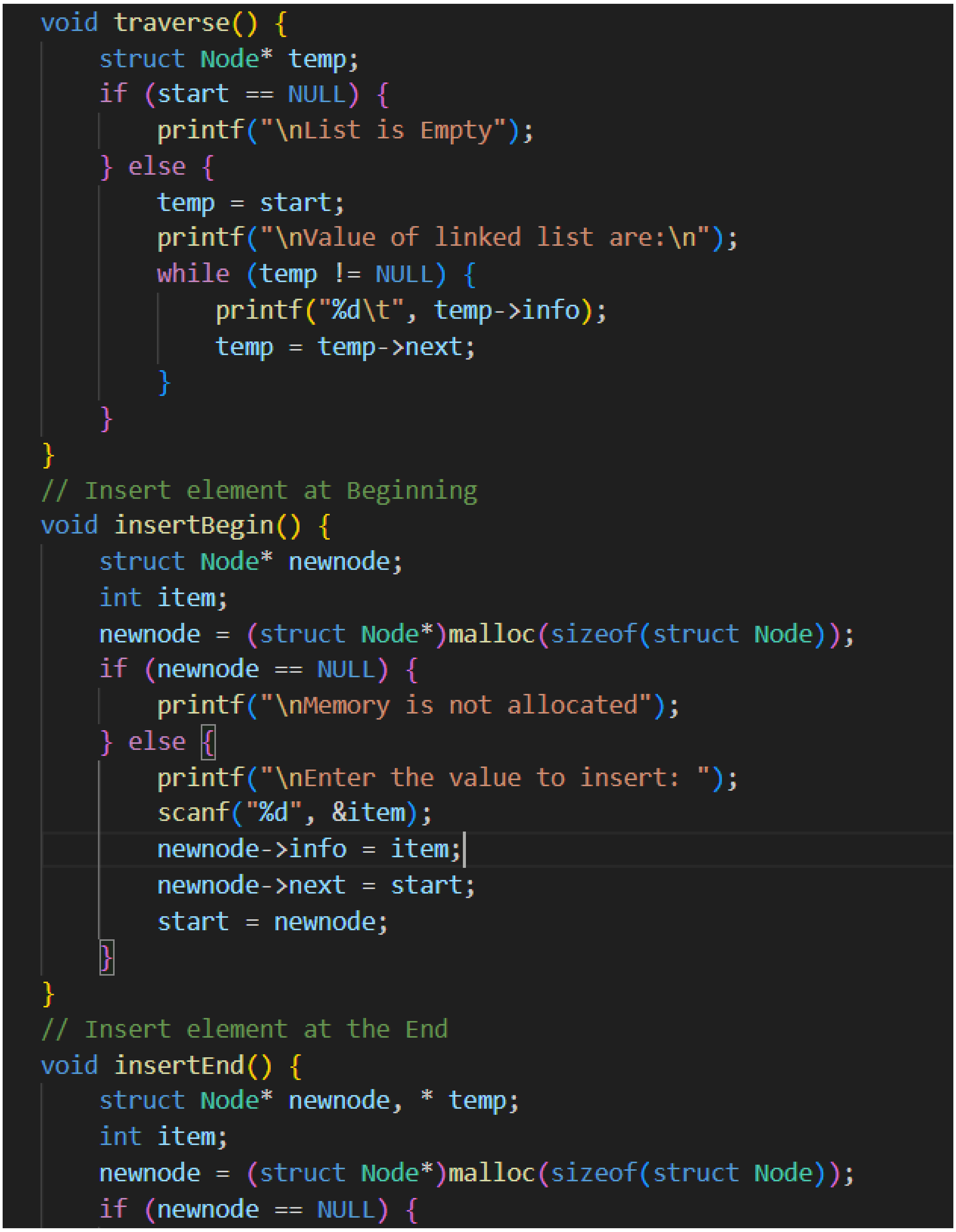
Question 1:

Write a program to perform all insertion technique in a linear linked list.

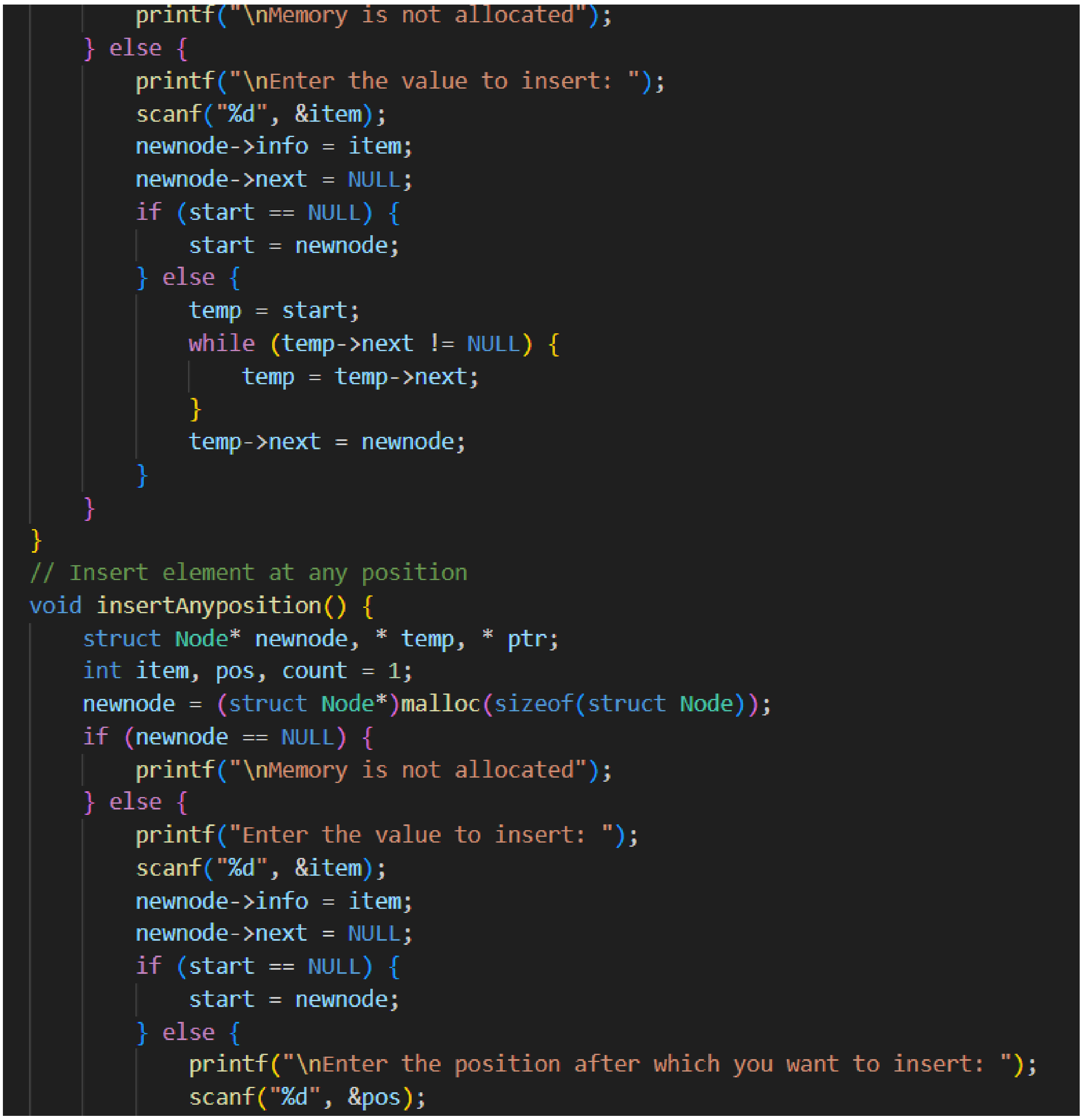
Source code:



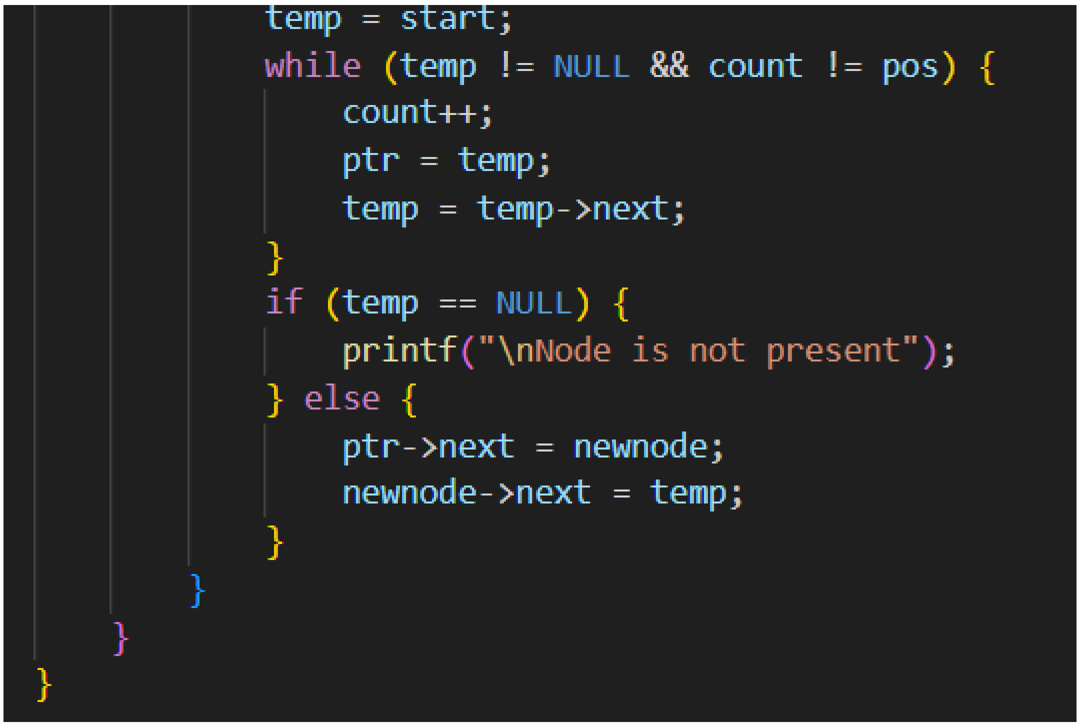
```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int info;
    struct Node* next;
};
struct Node* start = NULL;
void traverse();
void insertBegin();
void insertEnd();
void insertAnyposition();
int main() {
    int choice;
    while (1) {
        printf("\n\n1.Insert at Beginning\n2.Traverse\n3.Insert at End\n4.Insert any Position\n5.Exit\n");
        printf("\nEnter the choice: ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            insertBegin();
            break;
        case 2:
            traverse();
            break;
        case 3:
            insertEnd();
            break;
        case 4:
            insertAnyposition();
            break;
        case 5:
            exit(0);
        default:
            printf("\nWrong choice\n");
        }
    }
}
```



```c
void traverse() {
    struct Node* temp;
    if (start == NULL) {
        printf("\nList is Empty");
    } else {
        temp = start;
        printf("\nValue of linked list are:\n");
        while (temp != NULL) {
            printf("%d\t", temp->info);
            temp = temp->next;
        }
    }
}
// Insert element at Beginning
void insertBegin() {
    struct Node* newnode;
    int item;
    newnode = (struct Node*)malloc(sizeof(struct Node));
    if (newnode == NULL) {
        printf("\nMemory is not allocated");
    } else {
        printf("\nEnter the value to insert: ");
        scanf("%d", &item);
        newnode->info = item;
        newnode->next = start;
        start = newnode;
    }
}
// Insert element at the End
void insertEnd() {
    struct Node* newnode, * temp;
    int item;
    newnode = (struct Node*)malloc(sizeof(struct Node));
    if (newnode == NULL) {
```

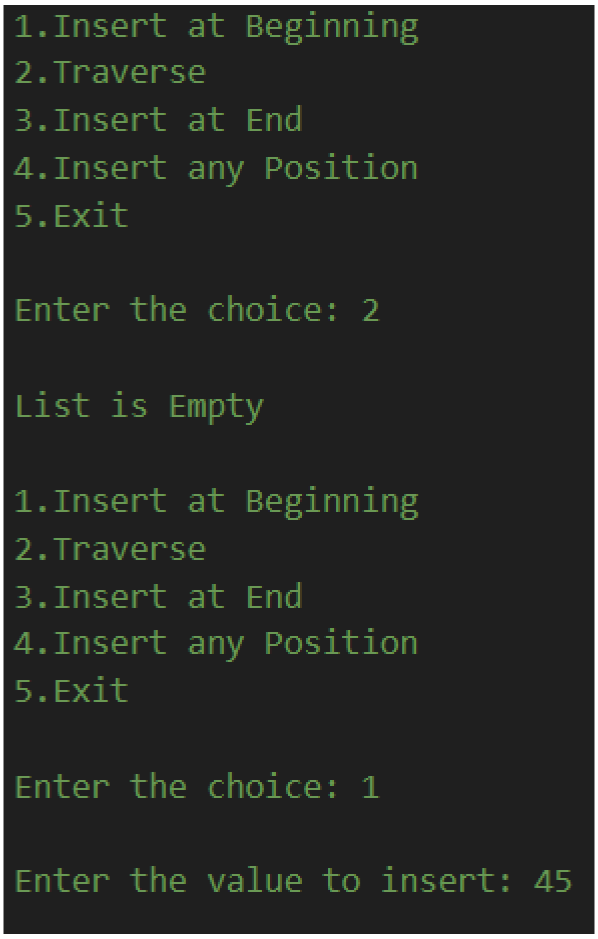


```
        printf("\nMemory is not allocated");
    } else {
        printf("\nEnter the value to insert: ");
        scanf("%d", &item);
        newnode->info = item;
        newnode->next = NULL;
        if (start == NULL) {
            start = newnode;
        } else {
            temp = start;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newnode;
        }
    }
}
// Insert element at any position
void insertAnyposition() {
    struct Node* newnode, * temp, * ptr;
    int item, pos, count = 1;
    newnode = (struct Node*)malloc(sizeof(struct Node));
    if (newnode == NULL) {
        printf("\nMemory is not allocated");
    } else {
        printf("Enter the value to insert: ");
        scanf("%d", &item);
        newnode->info = item;
        newnode->next = NULL;
        if (start == NULL) {
            start = newnode;
        } else {
            printf("\nEnter the position after which you want to insert: ");
            scanf("%d", &pos);
```

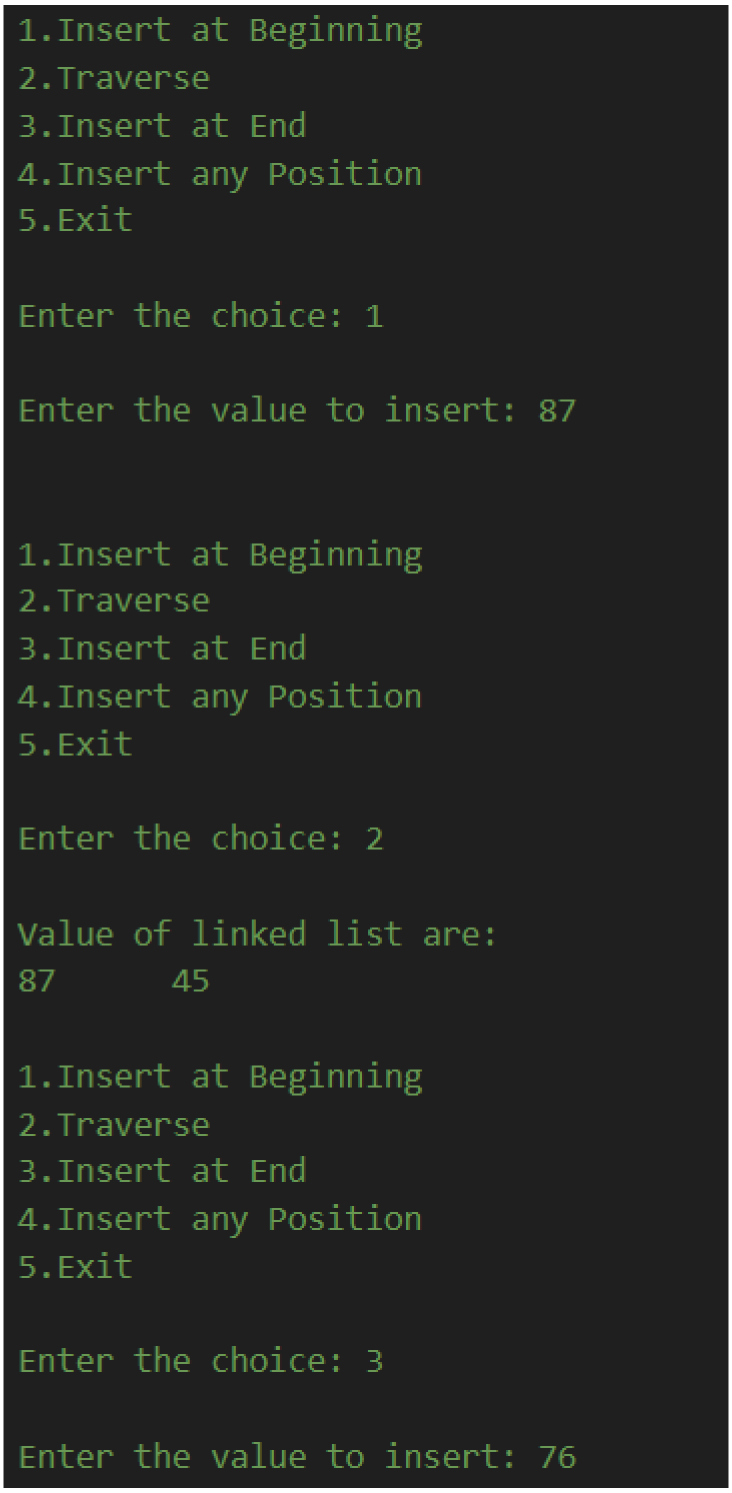


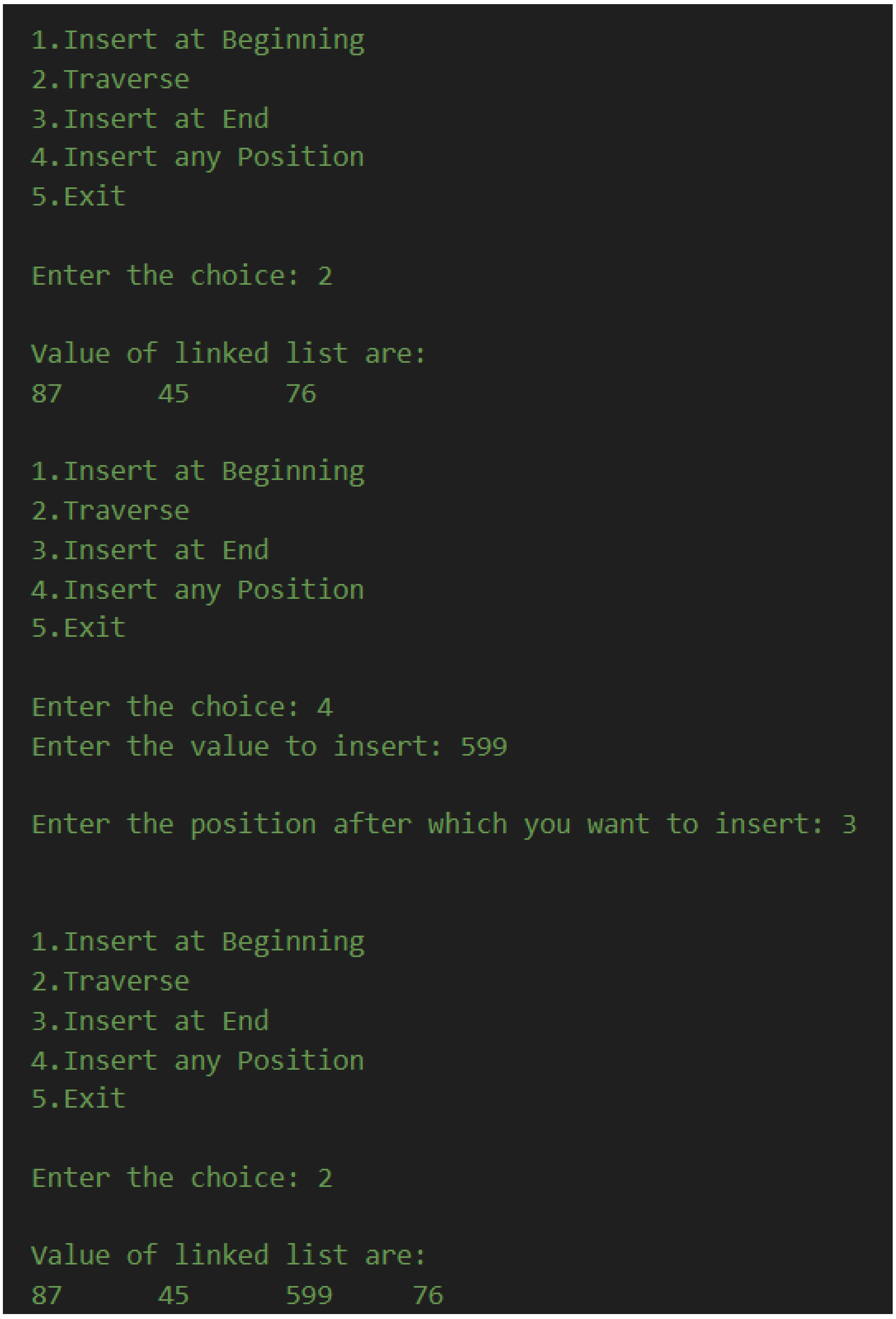
```c
            temp = start;
            while (temp != NULL && count != pos) {
                count++;
                ptr = temp;
                temp = temp->next;
            }
            if (temp == NULL) {
                printf("\nNode is not present");
            } else {
                ptr->next = newnode;
                newnode->next = temp;
            }
        }
    }
}
```

Output:

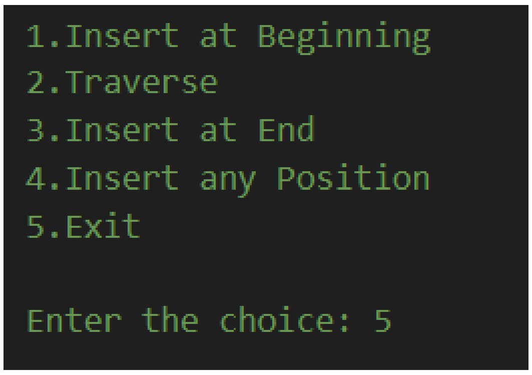


```
1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 2

List is Empty

1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 1

Enter the value to insert: 45
```


```
1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 1

Enter the value to insert: 87


1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 2

Value of linked list are:
87      45

1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 3

Enter the value to insert: 76
```



```
1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 2

Value of linked list are:
87      45      76

1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 4
Enter the value to insert: 599

Enter the position after which you want to insert: 3


1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 2

Value of linked list are:
87      45      599     76
```

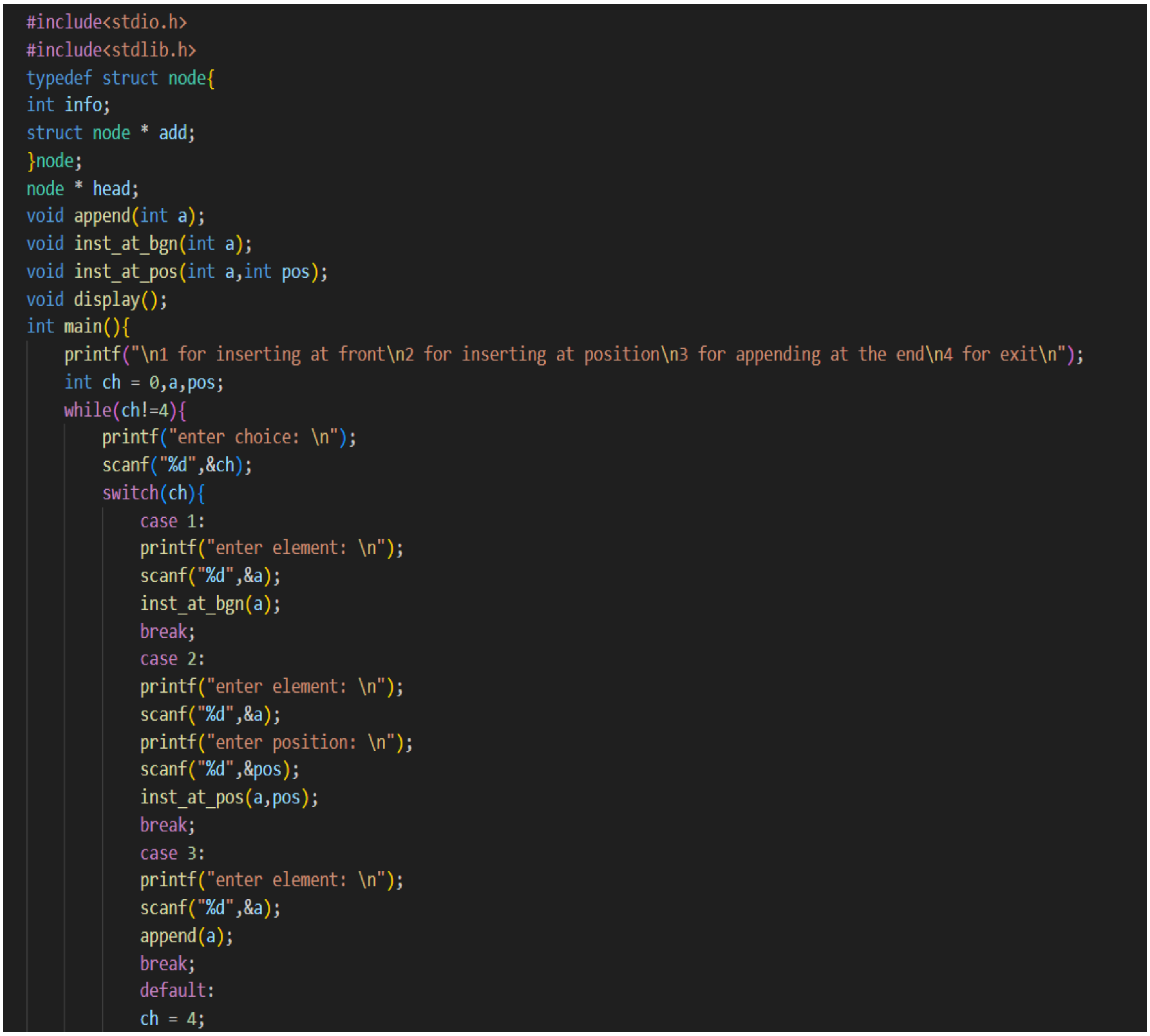


```
1.Insert at Beginning
2.Traverse
3.Insert at End
4.Insert any Position
5.Exit

Enter the choice: 5
```

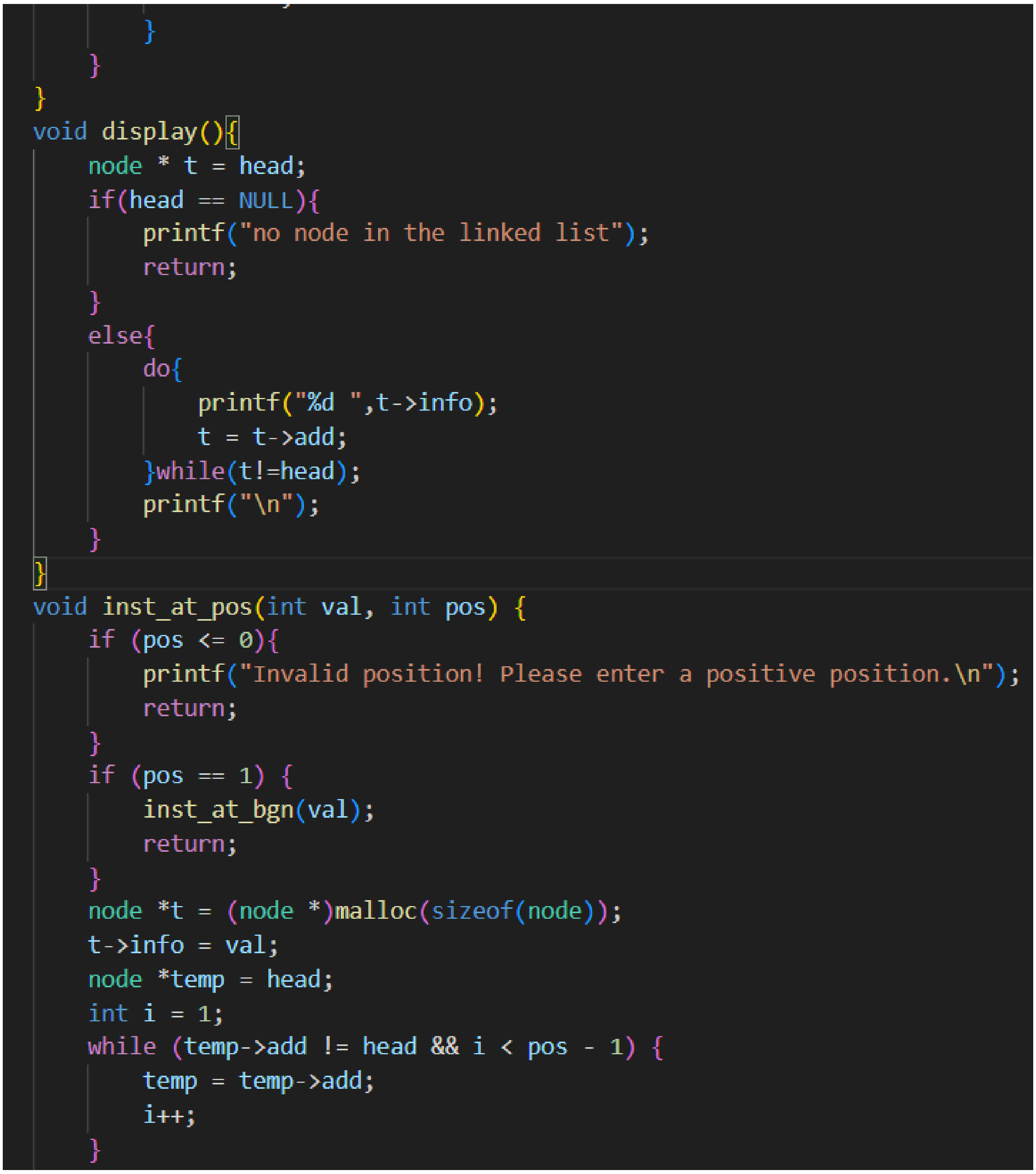
Question 2:

Write a program to perform all insertion technique in a circular linked list.

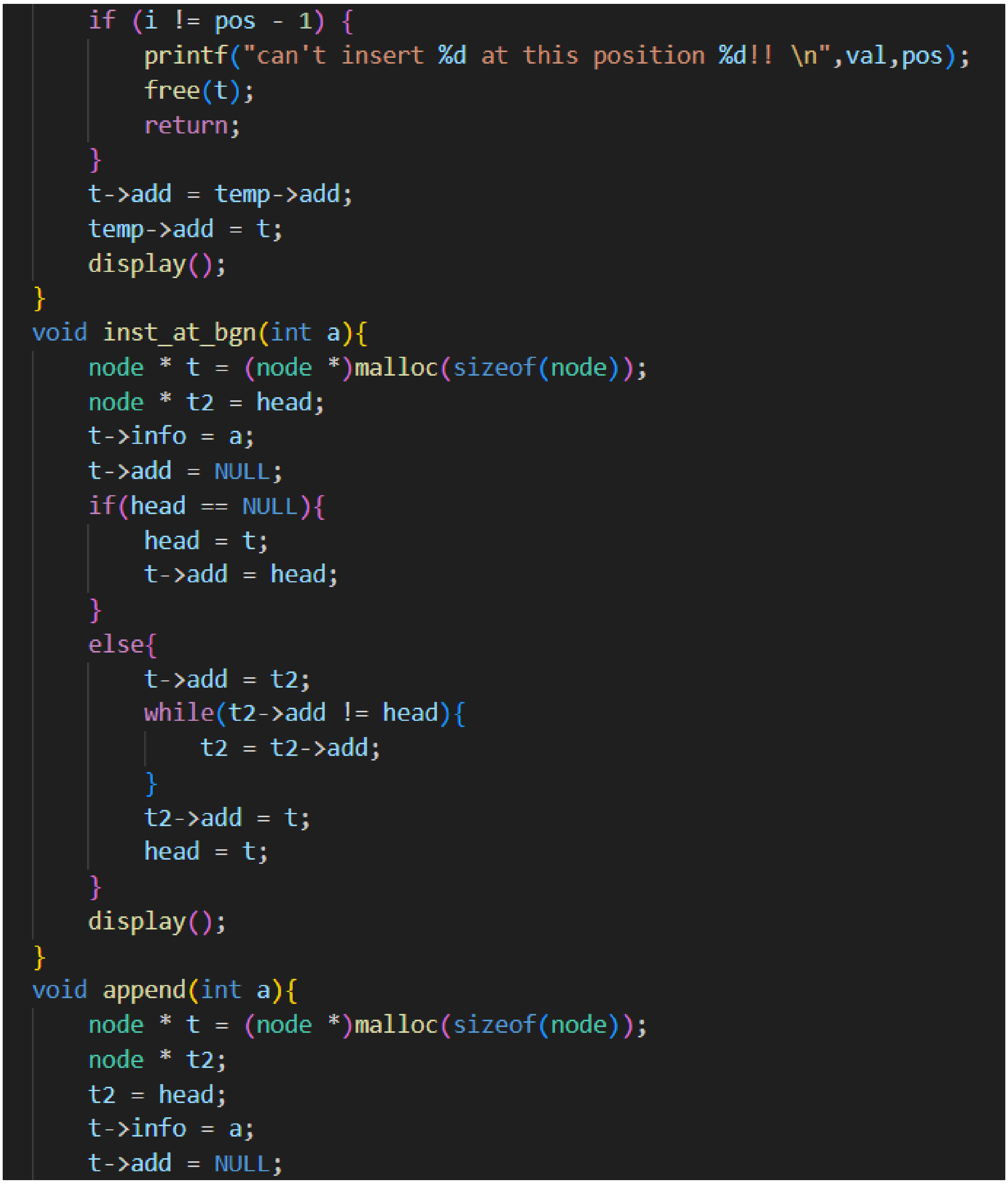
Source code:



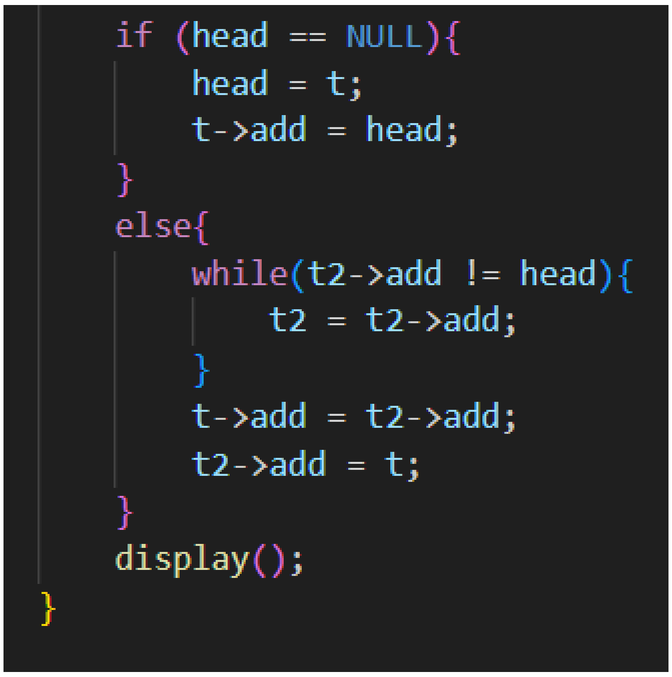
```c
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
int info;
struct node * add;
}node;
node * head;
void append(int a);
void inst_at_bgn(int a);
void inst_at_pos(int a,int pos);
void display();
int main(){
    printf("\n1 for inserting at front\n2 for inserting at position\n3 for appending at the end\n4 for exit\n");
    int ch = 0,a,pos;
    while(ch!=4){
        printf("enter choice: \n");
        scanf("%d",&ch);
        switch(ch){
            case 1:
            printf("enter element: \n");
            scanf("%d",&a);
            inst_at_bgn(a);
            break;
            case 2:
            printf("enter element: \n");
            scanf("%d",&a);
            printf("enter position: \n");
            scanf("%d",&pos);
            inst_at_pos(a,pos);
            break;
            case 3:
            printf("enter element: \n");
            scanf("%d",&a);
            append(a);
            break;
            default:
            ch = 4;
```



```
        }
    }
}
void display(){
    node * t = head;
    if(head == NULL){
        printf("no node in the linked list");
        return;
    }
    else{
        do{
            printf("%d ",t->info);
            t = t->add;
        }while(t!=head);
        printf("\n");
    }
}
void inst_at_pos(int val, int pos) {
    if (pos <= 0){
        printf("Invalid position! Please enter a positive position.\n");
        return;
    }
    if (pos == 1) {
        inst_at_bgn(val);
        return;
    }
    node *t = (node *)malloc(sizeof(node));
    t->info = val;
    node *temp = head;
    int i = 1;
    while (temp->add != head && i < pos - 1) {
        temp = temp->add;
        i++;
    }
```

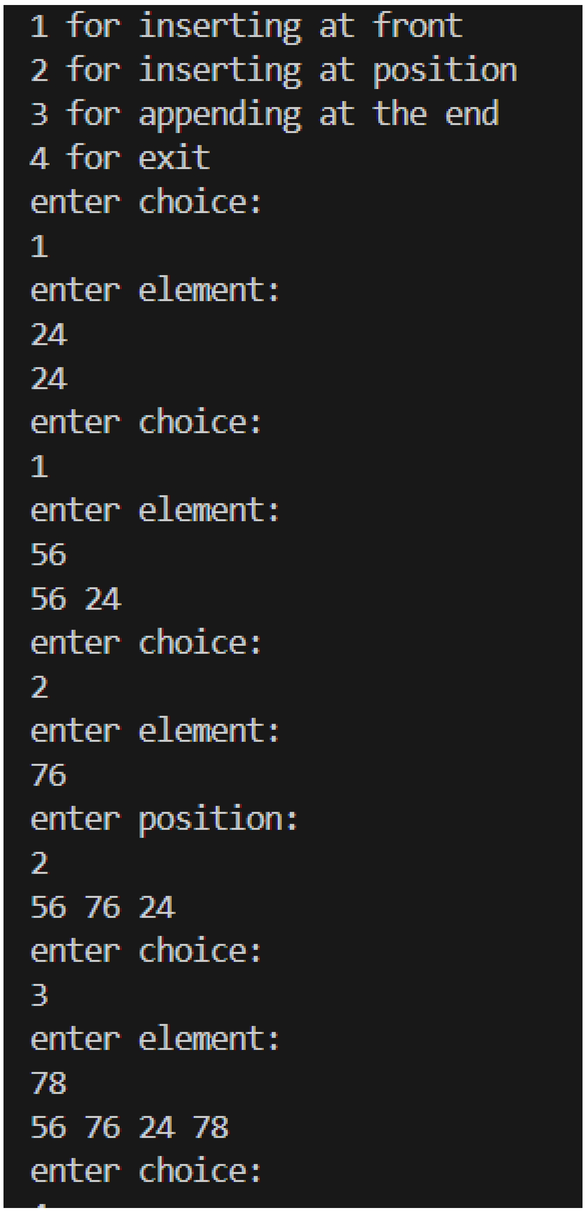


```
        if (i != pos - 1) {
            printf("can't insert %d at this position %d!! \n",val,pos);
            free(t);
            return;
        }
        t->add = temp->add;
        temp->add = t;
        display();
}
void inst_at_bgn(int a){
        node * t = (node *)malloc(sizeof(node));
        node * t2 = head;
        t->info = a;
        t->add = NULL;
        if(head == NULL){
            head = t;
            t->add = head;
        }
        else{
            t->add = t2;
            while(t2->add != head){
                t2 = t2->add;
            }
            t2->add = t;
            head = t;
        }
        display();
}
void append(int a){
        node * t = (node *)malloc(sizeof(node));
        node * t2;
        t2 = head;
        t->info = a;
        t->add = NULL;
```



```
    if (head == NULL){
        head = t;
        t->add = head;
    }
    else{
        while(t2->add != head){
            t2 = t2->add;
        }
        t->add = t2->add;
        t2->add = t;
    }
    display();
}
```

Output:

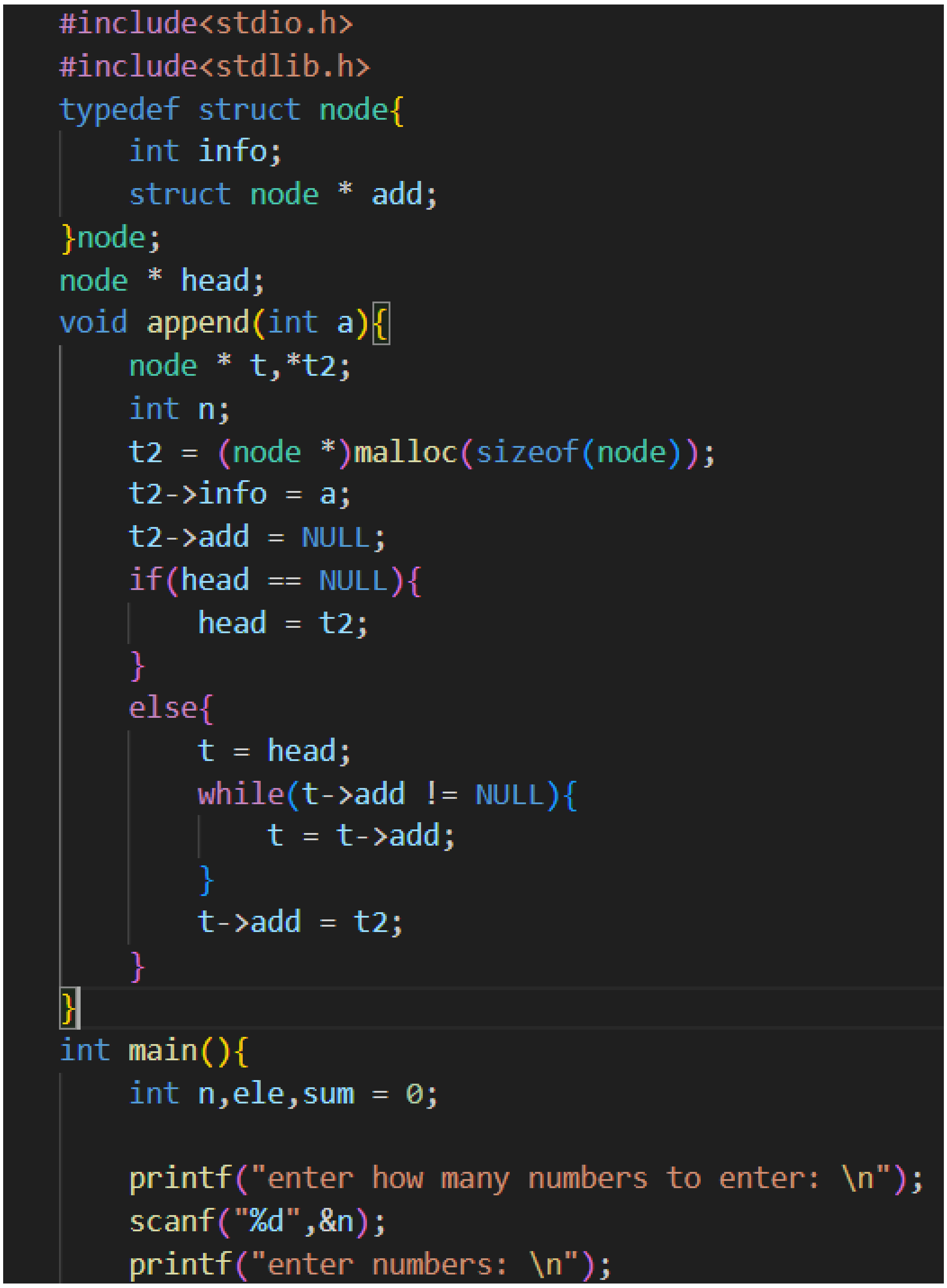


```
1 for inserting at front
2 for inserting at position
3 for appending at the end
4 for exit
enter choice:
1
enter element:
24
24
enter choice:
1
enter element:
56
56 24
enter choice:
2
enter element:
76
enter position:
2
56 76 24
enter choice:
3
enter element:
78
56 76 24 78
enter choice:
```

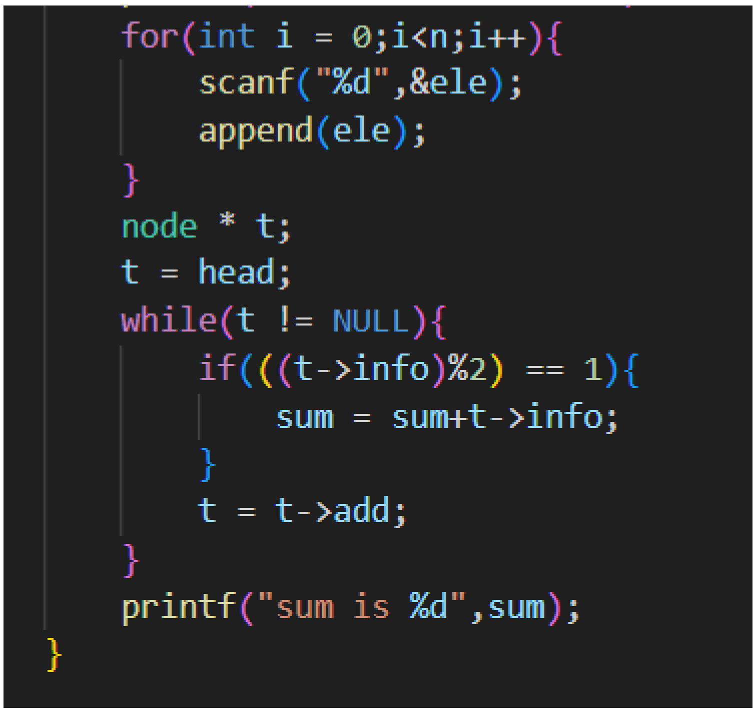
Question 3:

Write a program to find odd number sum in a linked list.

Source code:

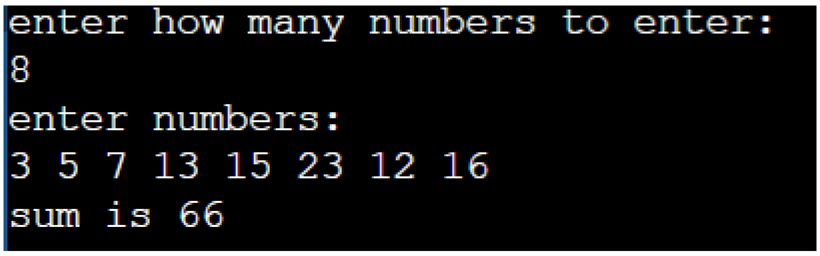


```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int info;
    struct node * add;
}node;
node * head;
void append(int a){
    node * t,*t2;
    int n;
    t2 = (node *)malloc(sizeof(node));
    t2->info = a;
    t2->add = NULL;
    if(head == NULL){
        head = t2;
    }
    else{
        t = head;
        while(t->add != NULL){
            t = t->add;
        }
        t->add = t2;
    }
}
int main(){
    int n,ele,sum = 0;

    printf("enter how many numbers to enter: \n");
    scanf("%d",&n);
    printf("enter numbers: \n");
```



```
    for(int i = 0;i<n;i++){
        scanf("%d",&ele);
        append(ele);
    }
    node * t;
    t = head;
    while(t != NULL){
        if(((t->info)%2) == 1){
            sum = sum+t->info;
        }
        t = t->add;
    }
    printf("sum is %d",sum);
}
```

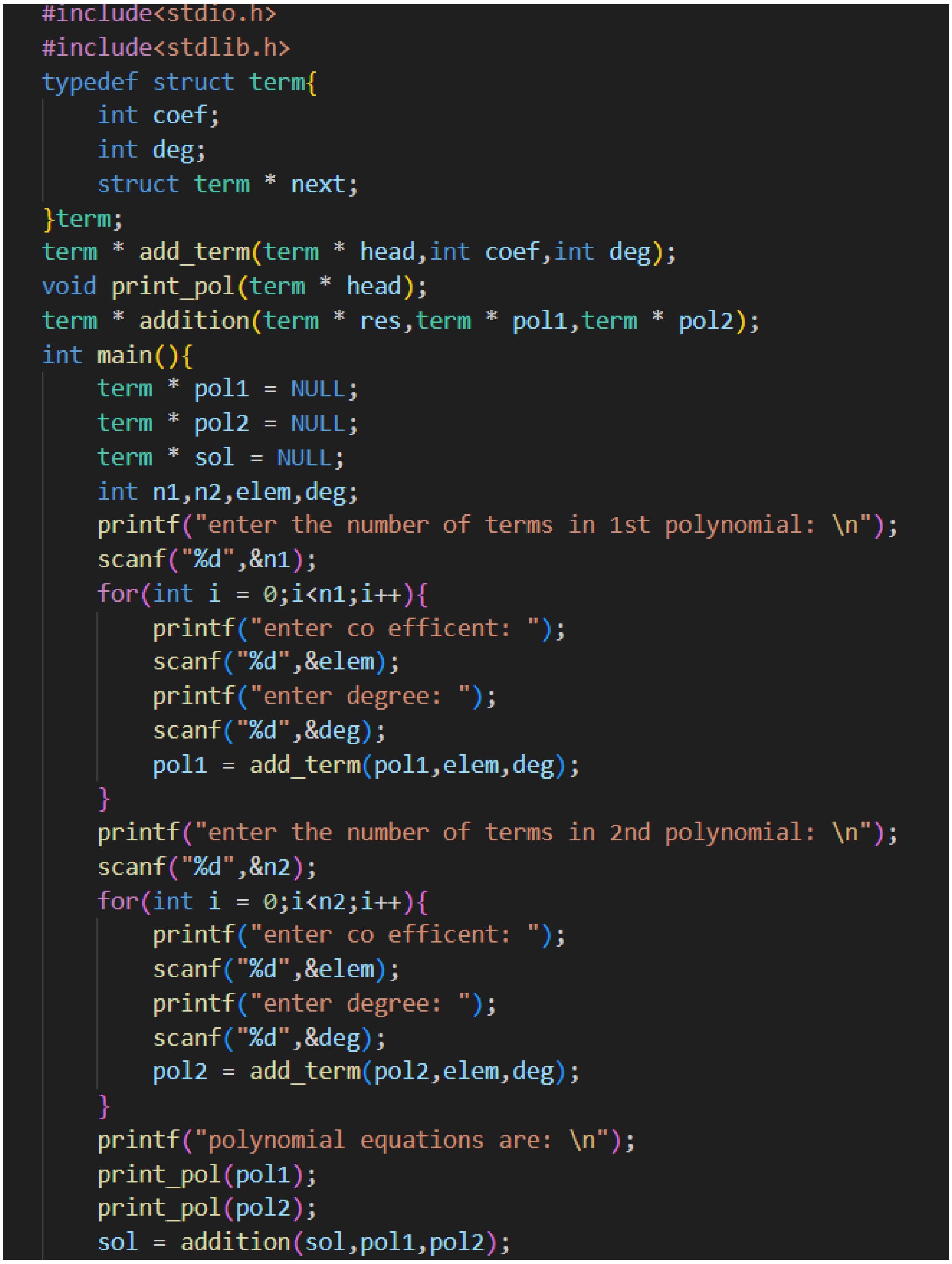
Output:



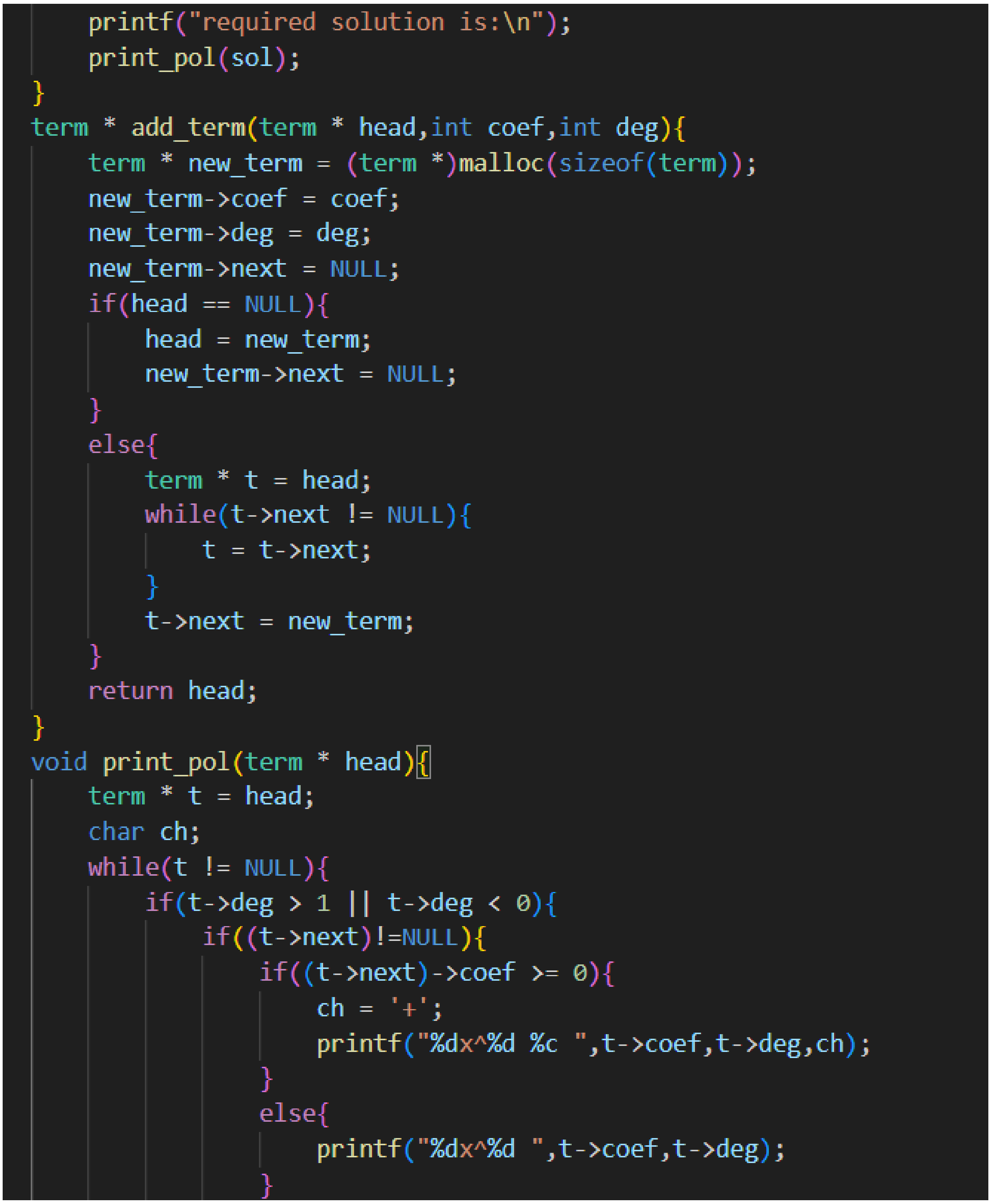
```
enter how many numbers to enter:
8
enter numbers:
3 5 7 13 15 23 12 16
sum is 66
```

4.Write a program to perform polynomial addition of two linked list.

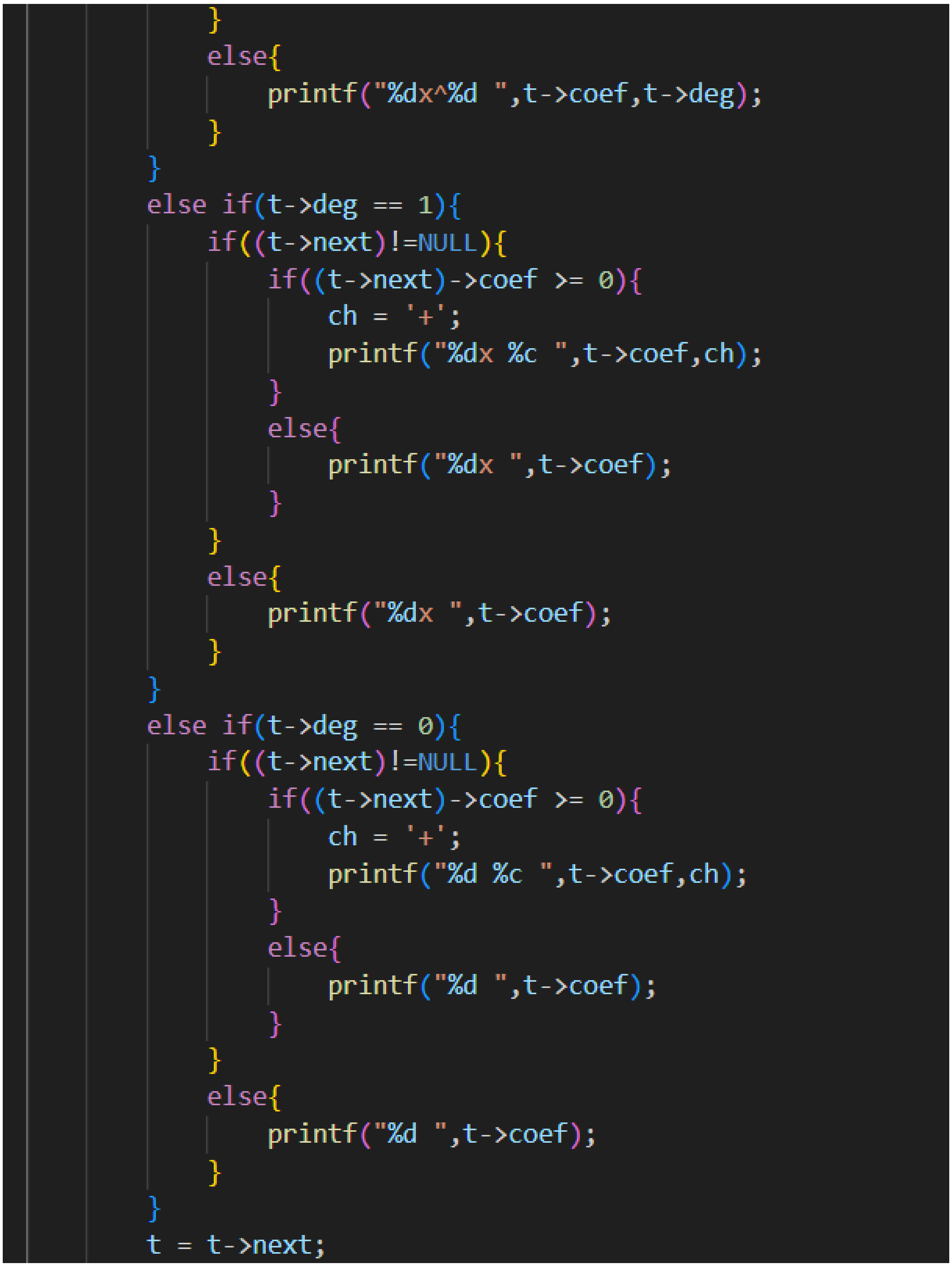
Source code:



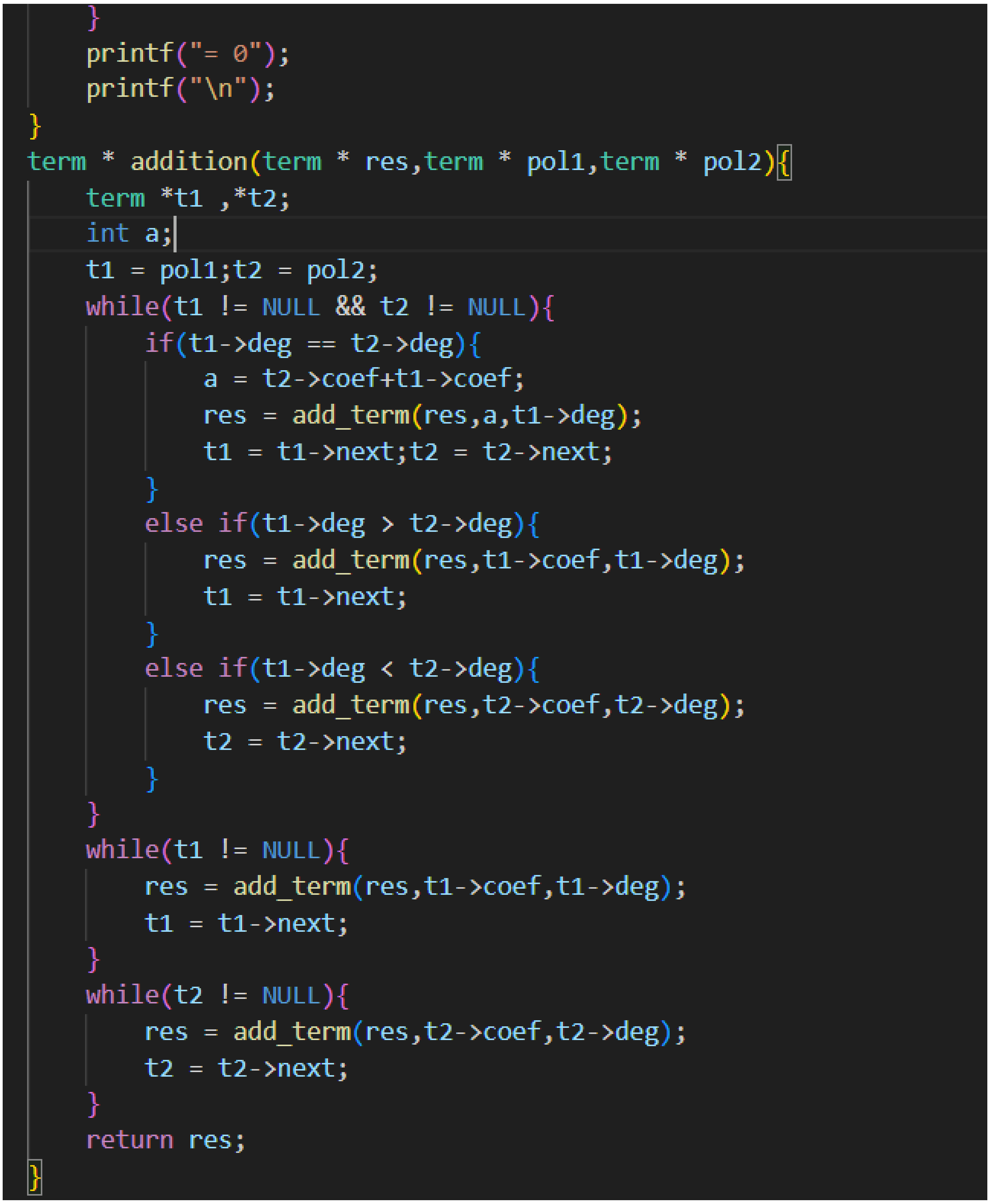
```c
#include<stdio.h>
#include<stdlib.h>
typedef struct term{
    int coef;
    int deg;
    struct term * next;
}term;
term * add_term(term * head,int coef,int deg);
void print_pol(term * head);
term * addition(term * res,term * pol1,term * pol2);
int main(){
    term * pol1 = NULL;
    term * pol2 = NULL;
    term * sol = NULL;
    int n1,n2,elem,deg;
    printf("enter the number of terms in 1st polynomial: \n");
    scanf("%d",&n1);
    for(int i = 0;i<n1;i++){
        printf("enter co efficent: ");
        scanf("%d",&elem);
        printf("enter degree: ");
        scanf("%d",&deg);
        pol1 = add_term(pol1,elem,deg);
    }
    printf("enter the number of terms in 2nd polynomial: \n");
    scanf("%d",&n2);
    for(int i = 0;i<n2;i++){
        printf("enter co efficent: ");
        scanf("%d",&elem);
        printf("enter degree: ");
        scanf("%d",&deg);
        pol2 = add_term(pol2,elem,deg);
    }
    printf("polynomial equations are: \n");
    print_pol(pol1);
    print_pol(pol2);
    sol = addition(sol,pol1,pol2);
```



```c
    printf("required solution is:\n");
    print_pol(sol);
}
term * add_term(term * head,int coef,int deg){
    term * new_term = (term *)malloc(sizeof(term));
    new_term->coef = coef;
    new_term->deg = deg;
    new_term->next = NULL;
    if(head == NULL){
        head = new_term;
        new_term->next = NULL;
    }
    else{
        term * t = head;
        while(t->next != NULL){
            t = t->next;
        }
        t->next = new_term;
    }
    return head;
}
void print_pol(term * head){
    term * t = head;
    char ch;
    while(t != NULL){
        if(t->deg > 1 || t->deg < 0){
            if((t->next)!=NULL){
                if((t->next)->coef >= 0){
                    ch = '+';
                    printf("%dx^%d %c ",t->coef,t->deg,ch);
                }
                else{
                    printf("%dx^%d ",t->coef,t->deg);
                }
```

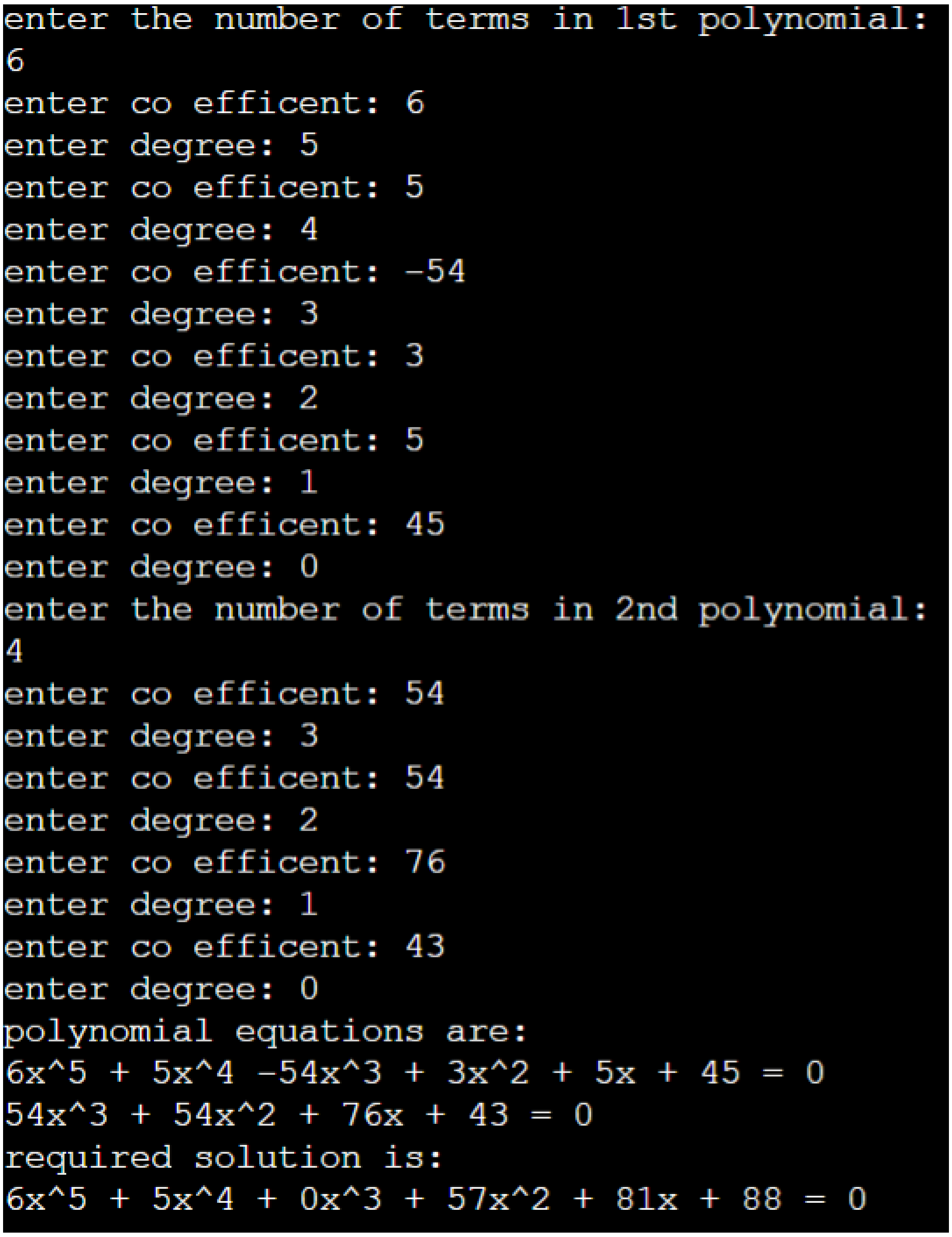


```c
            }
            else{
                printf("%dx^%d ",t->coef,t->deg);
            }
        }
        else if(t->deg == 1){
            if((t->next)!=NULL){
                if((t->next)->coef >= 0){
                    ch = '+';
                    printf("%dx %c ",t->coef,ch);
                }
                else{
                    printf("%dx ",t->coef);
                }
            }
            else{
                printf("%dx ",t->coef);
            }
        }
        else if(t->deg == 0){
            if((t->next)!=NULL){
                if((t->next)->coef >= 0){
                    ch = '+';
                    printf("%d %c ",t->coef,ch);
                }
                else{
                    printf("%d ",t->coef);
                }
            }
            else{
                printf("%d ",t->coef);
            }
        }
        t = t->next;
```



```c
    }
    printf("= 0");
    printf("\n");
}
term * addition(term * res,term * pol1,term * pol2){
    term *t1 ,*t2;
    int a;
    t1 = pol1;t2 = pol2;
    while(t1 != NULL && t2 != NULL){
        if(t1->deg == t2->deg){
            a = t2->coef+t1->coef;
            res = add_term(res,a,t1->deg);
            t1 = t1->next;t2 = t2->next;
        }
        else if(t1->deg > t2->deg){
            res = add_term(res,t1->coef,t1->deg);
            t1 = t1->next;
        }
        else if(t1->deg < t2->deg){
            res = add_term(res,t2->coef,t2->deg);
            t2 = t2->next;
        }
    }
    while(t1 != NULL){
        res = add_term(res,t1->coef,t1->deg);
        t1 = t1->next;
    }
    while(t2 != NULL){
        res = add_term(res,t2->coef,t2->deg);
        t2 = t2->next;
    }
    return res;
}
```

Output:



```
enter the number of terms in 1st polynomial:
6
enter co efficent: 6
enter degree: 5
enter co efficent: 5
enter degree: 4
enter co efficent: -54
enter degree: 3
enter co efficent: 3
enter degree: 2
enter co efficent: 5
enter degree: 1
enter co efficent: 45
enter degree: 0
enter the number of terms in 2nd polynomial:
4
enter co efficent: 54
enter degree: 3
enter co efficent: 54
enter degree: 2
enter co efficent: 76
enter degree: 1
enter co efficent: 43
enter degree: 0
polynomial equations are:
6x^5 + 5x^4 -54x^3 + 3x^2 + 5x + 45 = 0
54x^3 + 54x^2 + 76x + 43 = 0
required solution is:
6x^5 + 5x^4 + 0x^3 + 57x^2 + 81x + 88 = 0
```