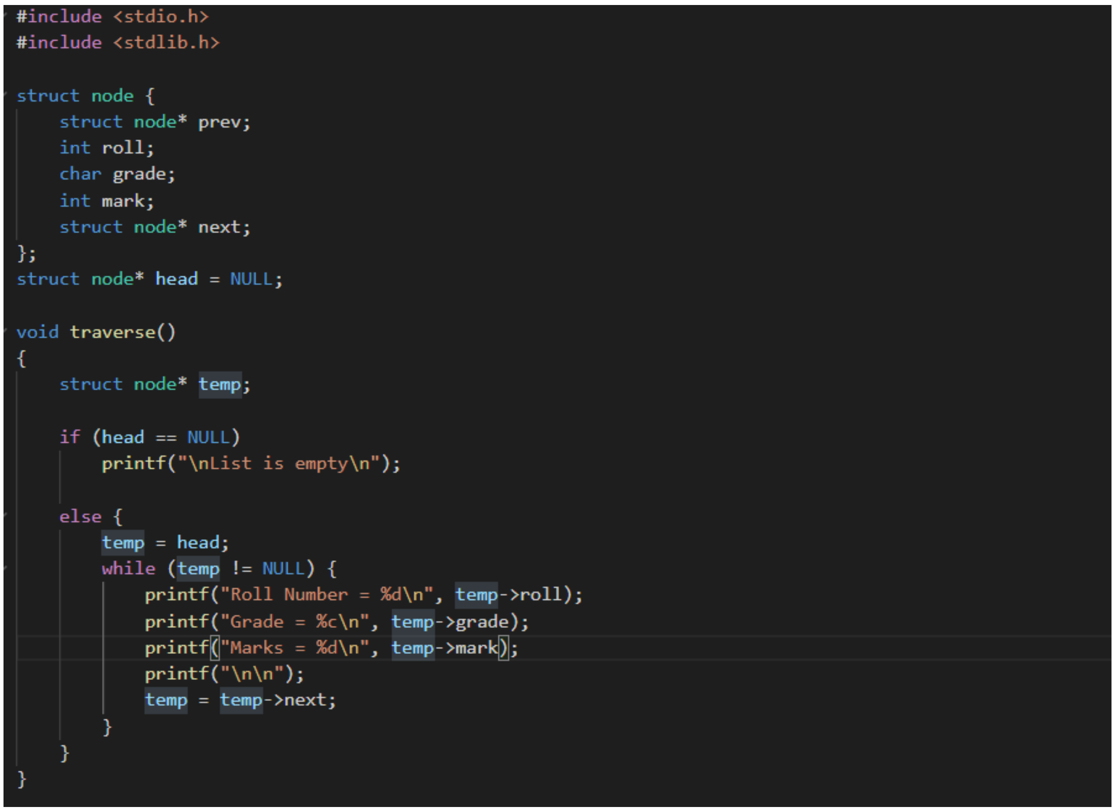
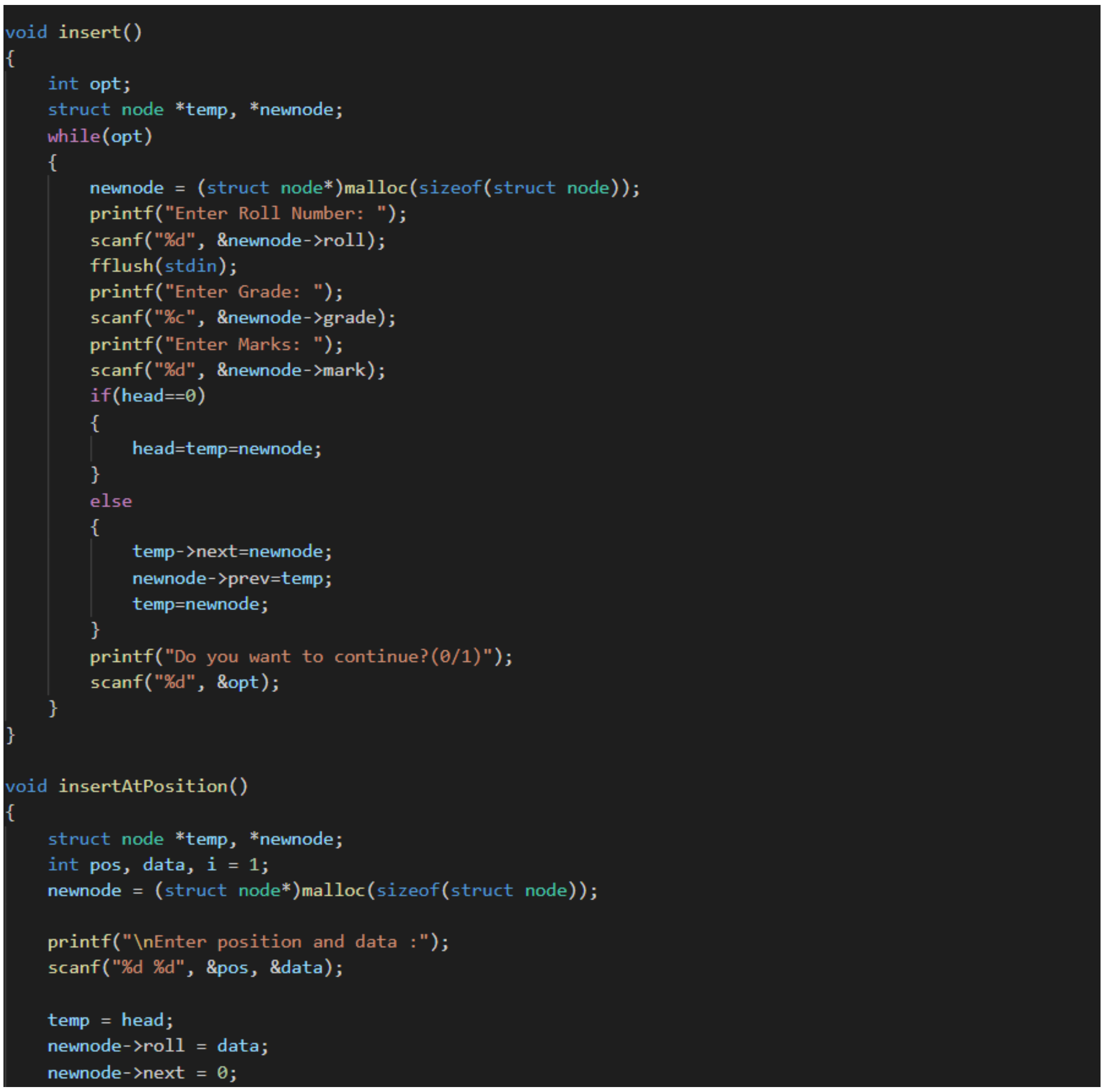
AGNIM GUPTA
2028083
A-23
CSSE

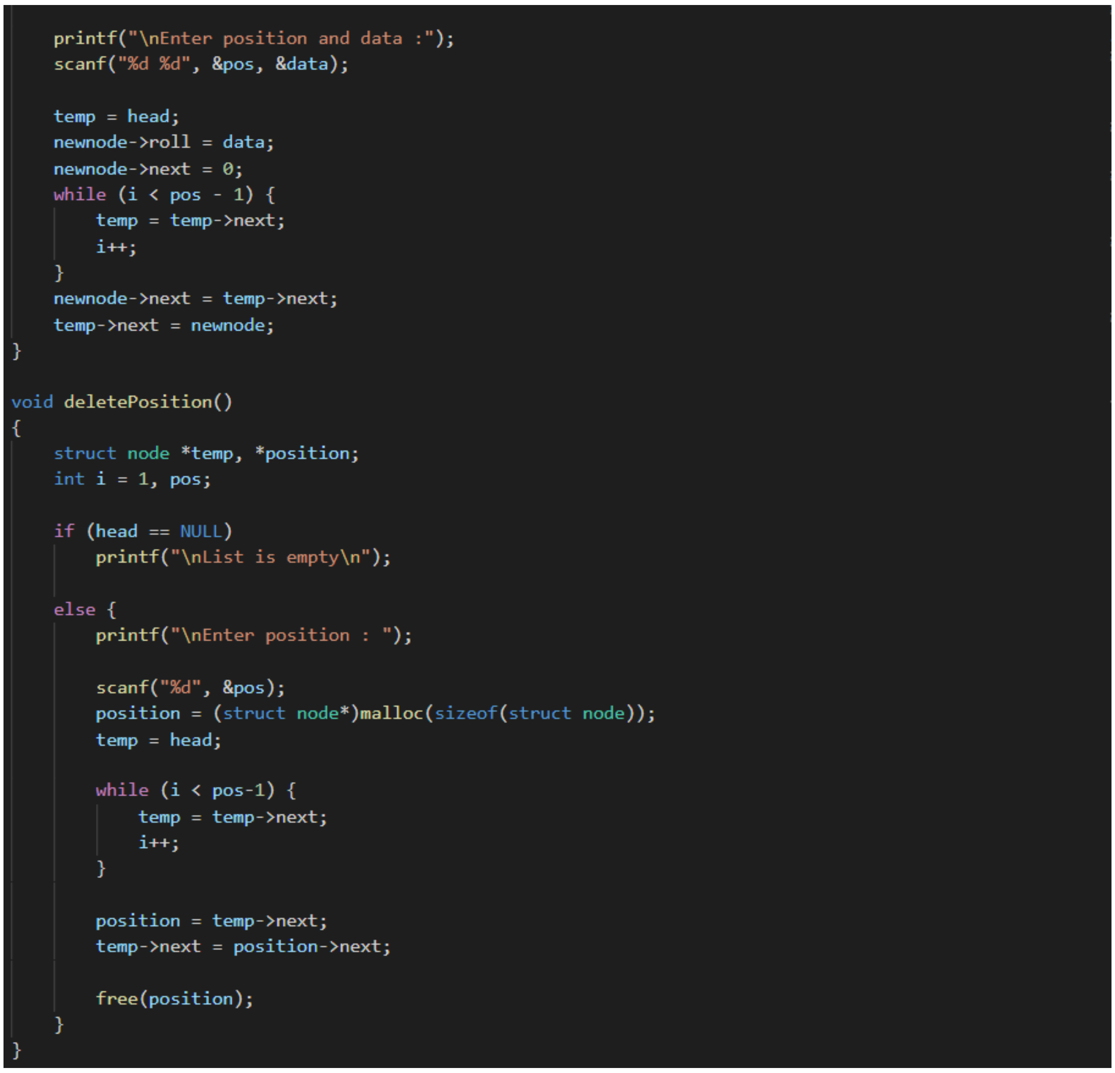
QUESTION 1



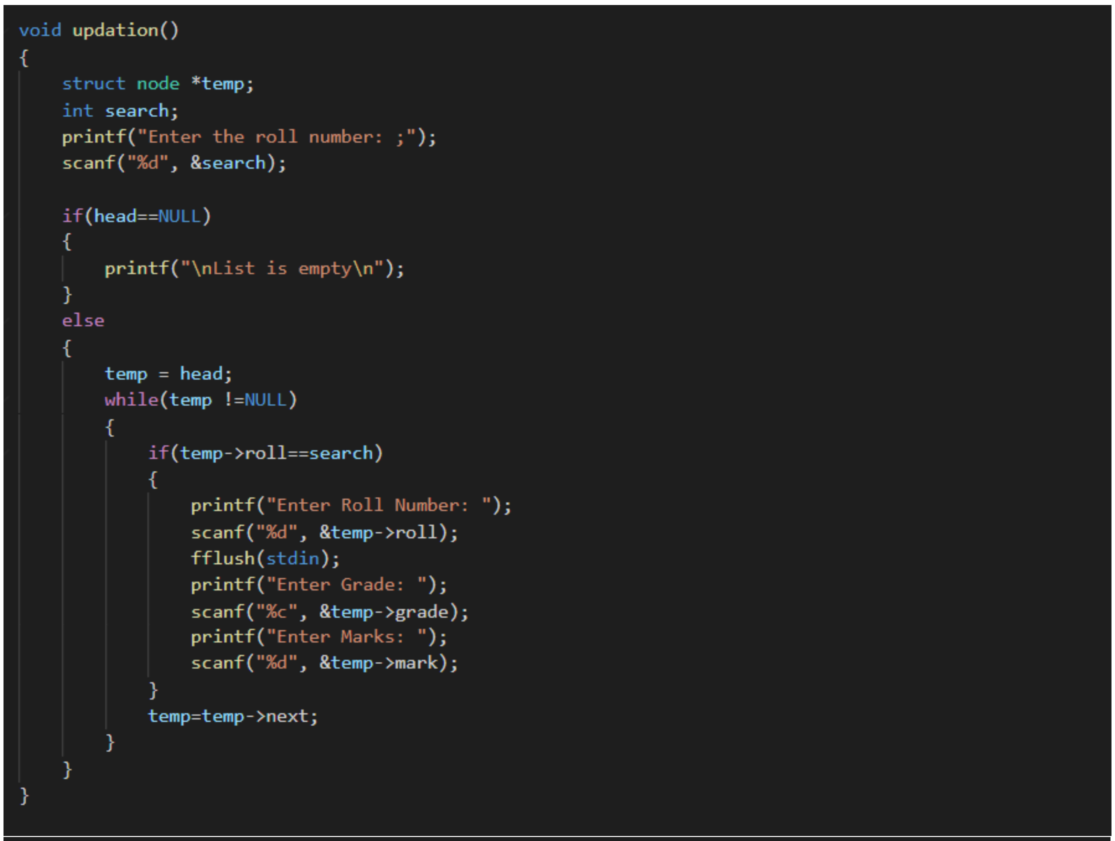
```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    struct node* prev;
    int roll;
    char grade;
    int mark;
    struct node* next;
};
struct node* head = NULL;

void traverse()
{
    struct node* temp;

    if (head == NULL)
        printf("\nList is empty\n");

    else {
        temp = head;
        while (temp != NULL) {
            printf("Roll Number = %d\n", temp->roll);
            printf("Grade = %c\n", temp->grade);
            printf("Marks = %d\n", temp->mark);
            printf("\n\n");
            temp = temp->next;
        }
    }
}
```



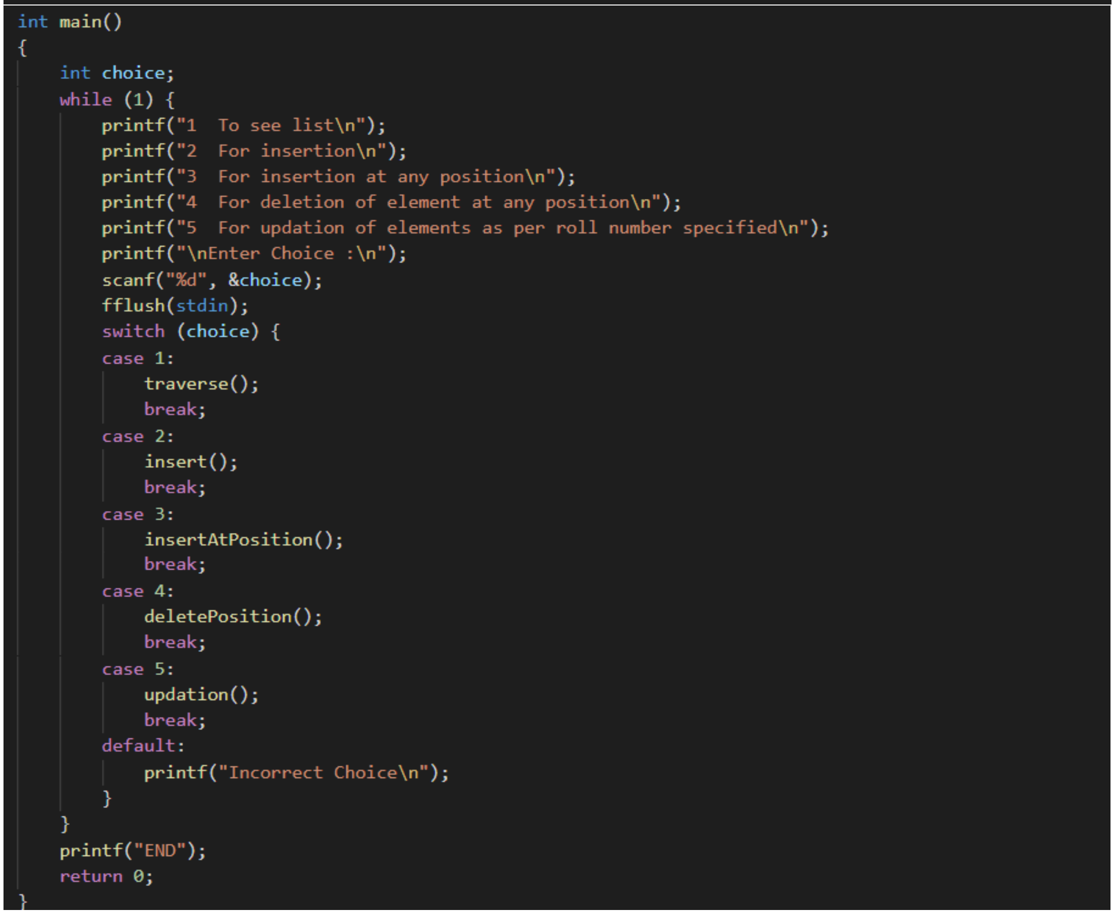
```c
void insert()
{
    int opt;
    struct node *temp, *newnode;
    while(opt)
    {
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter Roll Number: ");
        scanf("%d", &newnode->roll);
        fflush(stdin);
        printf("Enter Grade: ");
        scanf("%c", &newnode->grade);
        printf("Enter Marks: ");
        scanf("%d", &newnode->mark);
        if(head==0)
        {
            head=temp=newnode;
        }
        else
        {
            temp->next=newnode;
            newnode->prev=temp;
            temp=newnode;
        }
        printf("Do you want to continue?(0/1)");
        scanf("%d", &opt);
    }
}

void insertAtPosition()
{
    struct node *temp, *newnode;
    int pos, data, i = 1;
    newnode = (struct node*)malloc(sizeof(struct node));

    printf("\nEnter position and data :");
    scanf("%d %d", &pos, &data);

    temp = head;
    newnode->roll = data;
    newnode->next = 0;
```



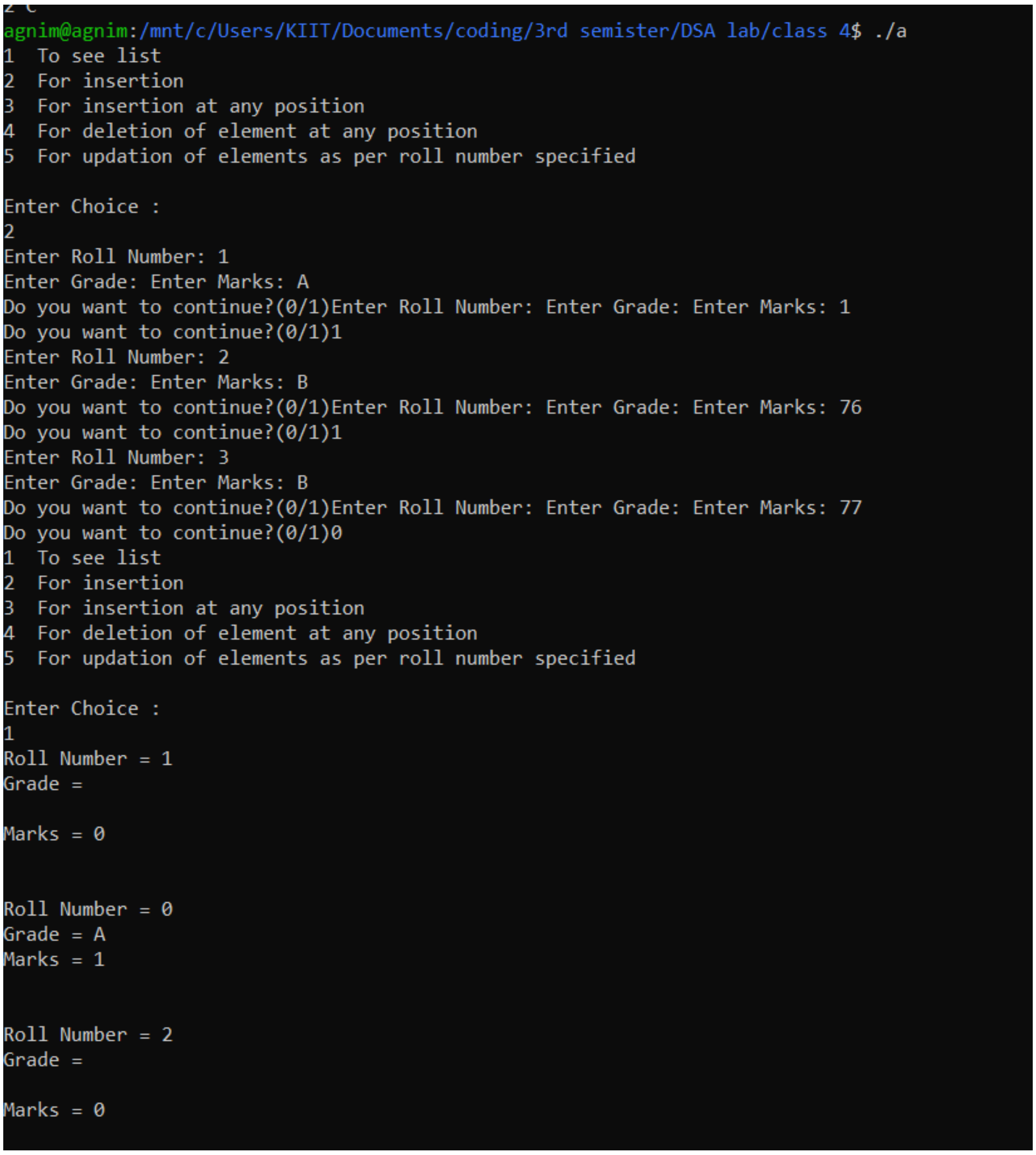
```c
    printf("\nEnter position and data :");
    scanf("%d %d", &pos, &data);

    temp = head;
    newnode->roll = data;
    newnode->next = 0;
    while (i < pos - 1) {
        temp = temp->next;
        i++;
    }
    newnode->next = temp->next;
    temp->next = newnode;
}

void deletePosition()
{
    struct node *temp, *position;
    int i = 1, pos;

    if (head == NULL)
        printf("\nList is empty\n");

    else {
        printf("\nEnter position : ");

        scanf("%d", &pos);
        position = (struct node*)malloc(sizeof(struct node));
        temp = head;

        while (i < pos-1) {
            temp = temp->next;
            i++;
        }

        position = temp->next;
        temp->next = position->next;

        free(position);
    }
}
```



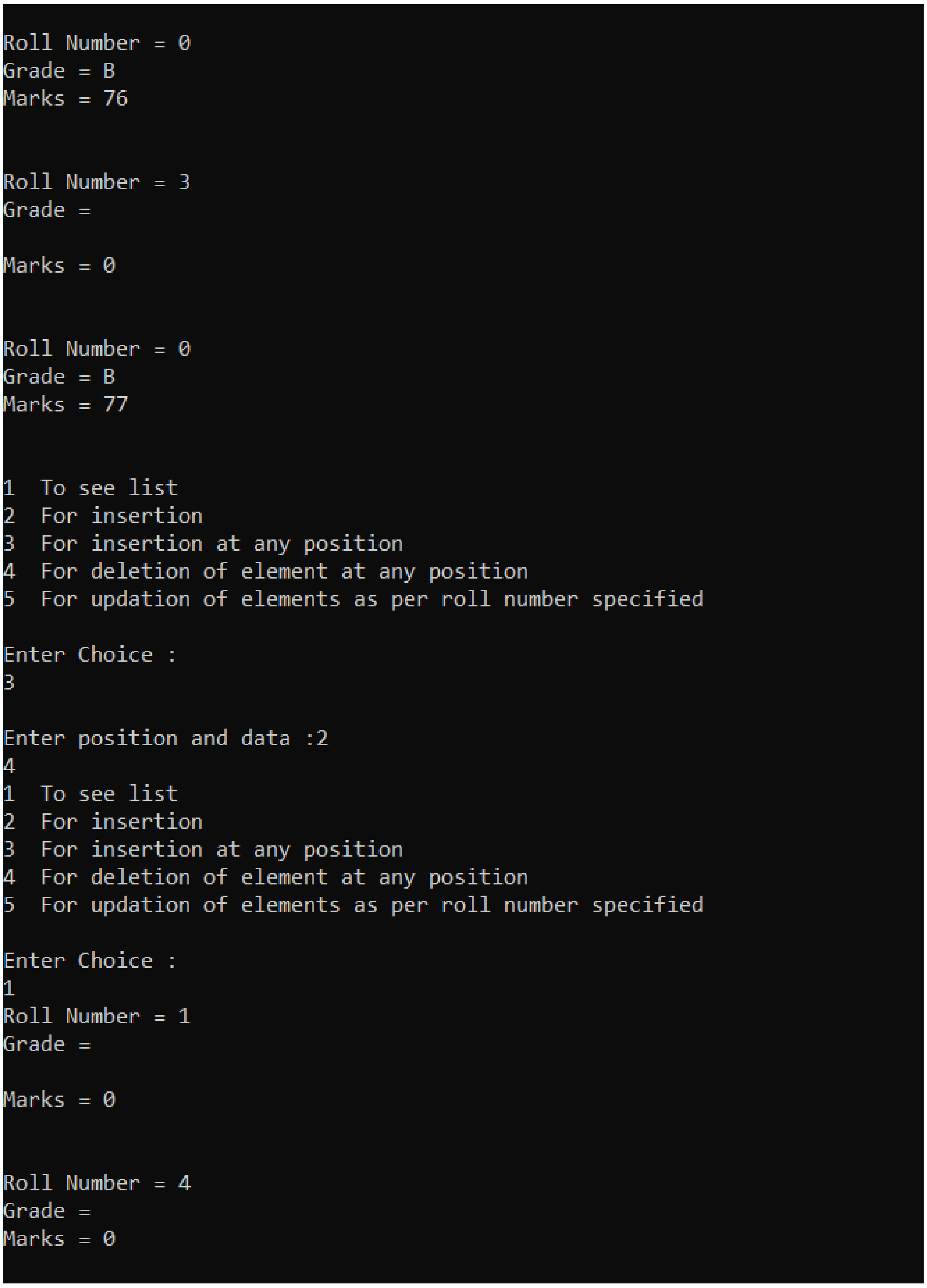
```c
void updation()
{
    struct node *temp;
    int search;
    printf("Enter the roll number: ;");
    scanf("%d", &search);

    if(head==NULL)
    {
        printf("\nList is empty\n");
    }
    else
    {
        temp = head;
        while(temp !=NULL)
        {
            if(temp->roll==search)
            {
                printf("Enter Roll Number: ");
                scanf("%d", &temp->roll);
                fflush(stdin);
                printf("Enter Grade: ");
                scanf("%c", &temp->grade);
                printf("Enter Marks: ");
                scanf("%d", &temp->mark);
            }
            temp=temp->next;
        }
    }
}
```

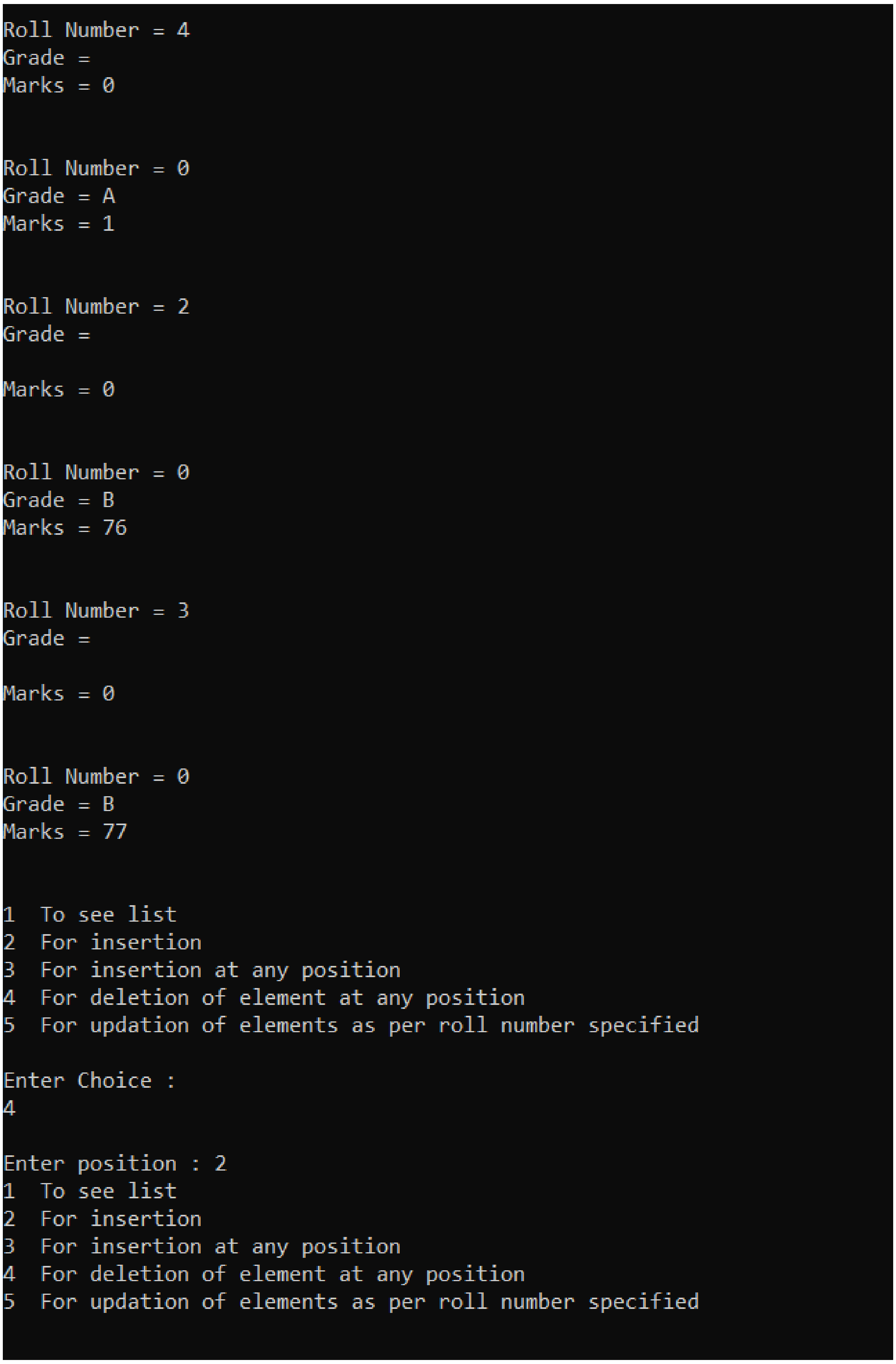


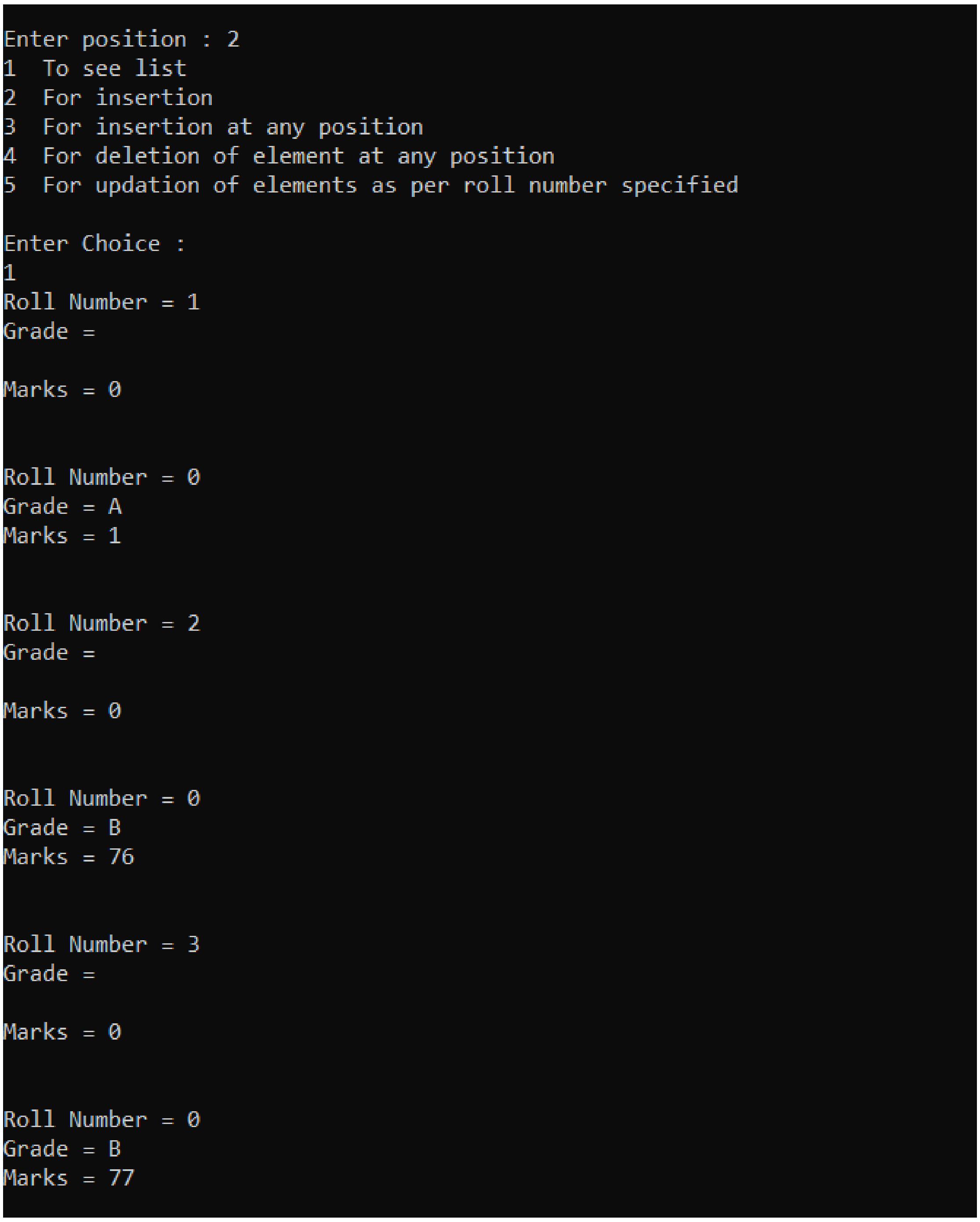
```c
int main()
{
    int choice;
    while (1) {
        printf("1  To see list\n");
        printf("2  For insertion\n");
        printf("3  For insertion at any position\n");
        printf("4  For deletion of element at any position\n");
        printf("5  For updation of elements as per roll number specified\n");
        printf("\nEnter Choice :\n");
        scanf("%d", &choice);
        fflush(stdin);
        switch (choice) {
        case 1:
            traverse();
            break;
        case 2:
            insert();
            break;
        case 3:
            insertAtPosition();
            break;
        case 4:
            deletePosition();
            break;
        case 5:
            updation();
            break;
        default:
            printf("Incorrect Choice\n");
        }
    }
    printf("END");
    return 0;
}
```



```
agnim@agnim:/mnt/c/Users/KIIT/Documents/coding/3rd semister/DSA lab/class 4$ ./a
1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
2
Enter Roll Number: 1
Enter Grade: Enter Marks: A
Do you want to continue?(0/1)Enter Roll Number: Enter Grade: Enter Marks: 1
Do you want to continue?(0/1)1
Enter Roll Number: 2
Enter Grade: Enter Marks: B
Do you want to continue?(0/1)Enter Roll Number: Enter Grade: Enter Marks: 76
Do you want to continue?(0/1)1
Enter Roll Number: 3
Enter Grade: Enter Marks: B
Do you want to continue?(0/1)Enter Roll Number: Enter Grade: Enter Marks: 77
Do you want to continue?(0/1)0
1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
1
Roll Number = 1
Grade =

Marks = 0


Roll Number = 0
Grade = A
Marks = 1


Roll Number = 2
Grade =

Marks = 0
```

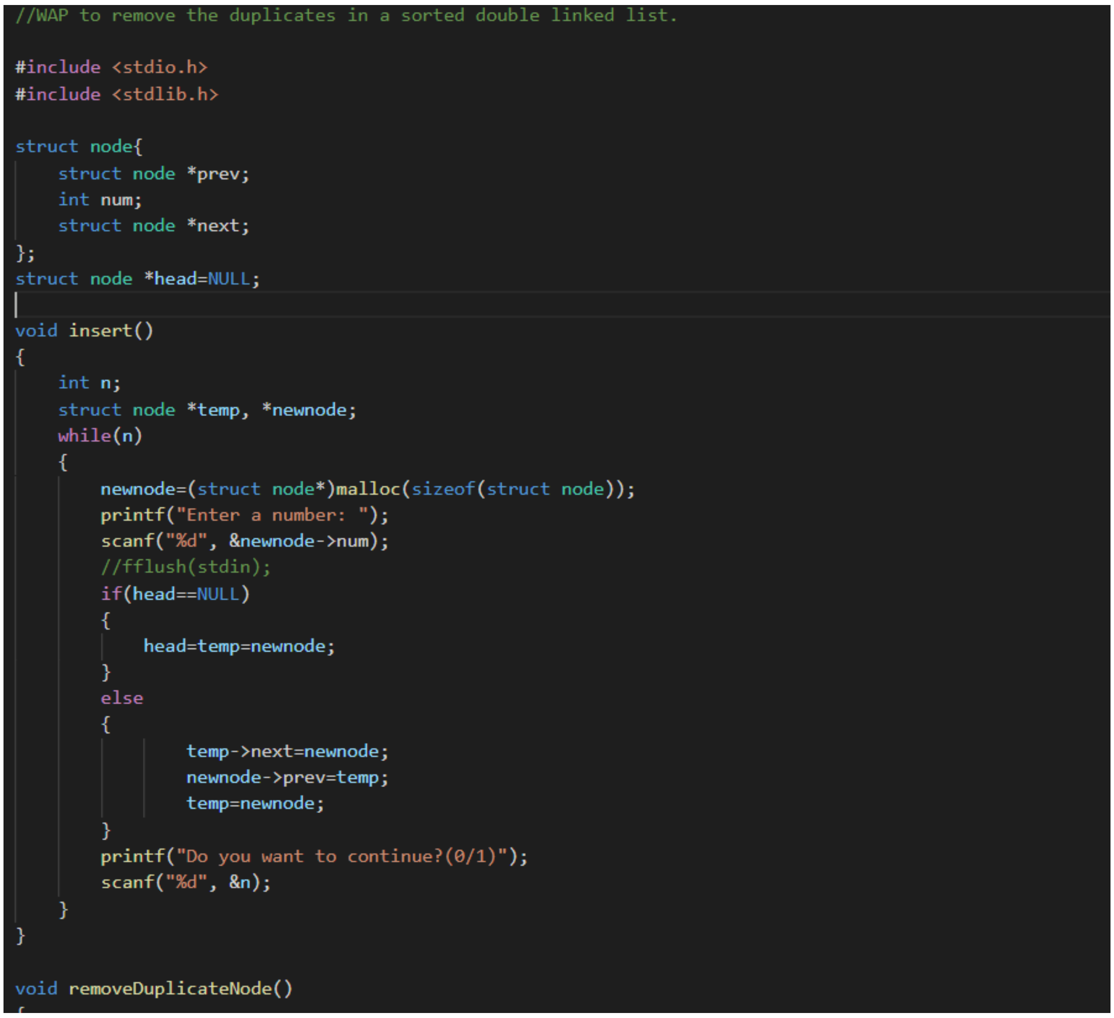

```
Roll Number = 0
Grade = B
Marks = 76


Roll Number = 3
Grade = 

Marks = 0


Roll Number = 0
Grade = B
Marks = 77


1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
3

Enter position and data :2
4
1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
1
Roll Number = 1
Grade = 

Marks = 0


Roll Number = 4
Grade = 
Marks = 0
```


```
Roll Number = 4
Grade =
Marks = 0


Roll Number = 0
Grade = A
Marks = 1


Roll Number = 2
Grade =

Marks = 0


Roll Number = 0
Grade = B
Marks = 76


Roll Number = 3
Grade =

Marks = 0


Roll Number = 0
Grade = B
Marks = 77


1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
4

Enter position : 2
1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified
```

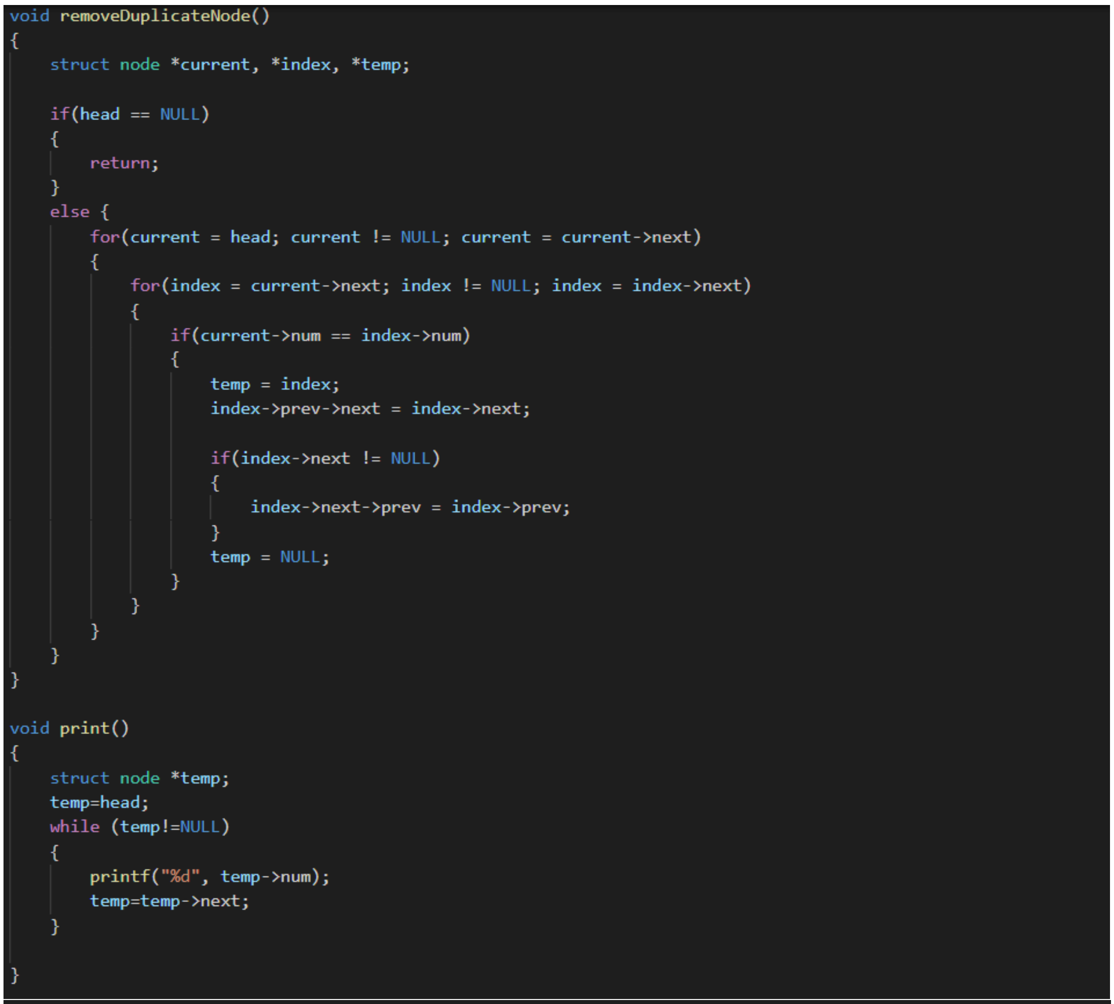


```
Enter position : 2
1  To see list
2  For insertion
3  For insertion at any position
4  For deletion of element at any position
5  For updation of elements as per roll number specified

Enter Choice :
1
Roll Number = 1
Grade =

Marks = 0


Roll Number = 0
Grade = A
Marks = 1


Roll Number = 2
Grade =

Marks = 0


Roll Number = 0
Grade = B
Marks = 76


Roll Number = 3
Grade =

Marks = 0


Roll Number = 0
Grade = B
Marks = 77
```

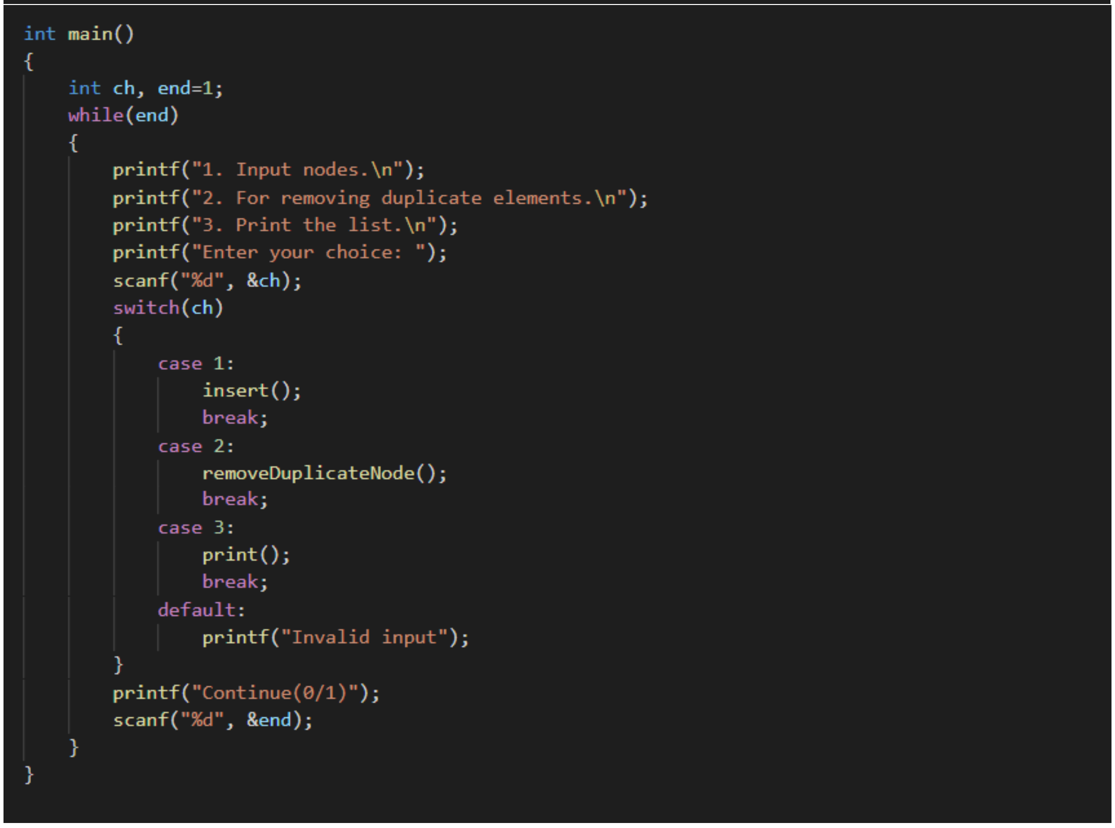
QUESTION 2



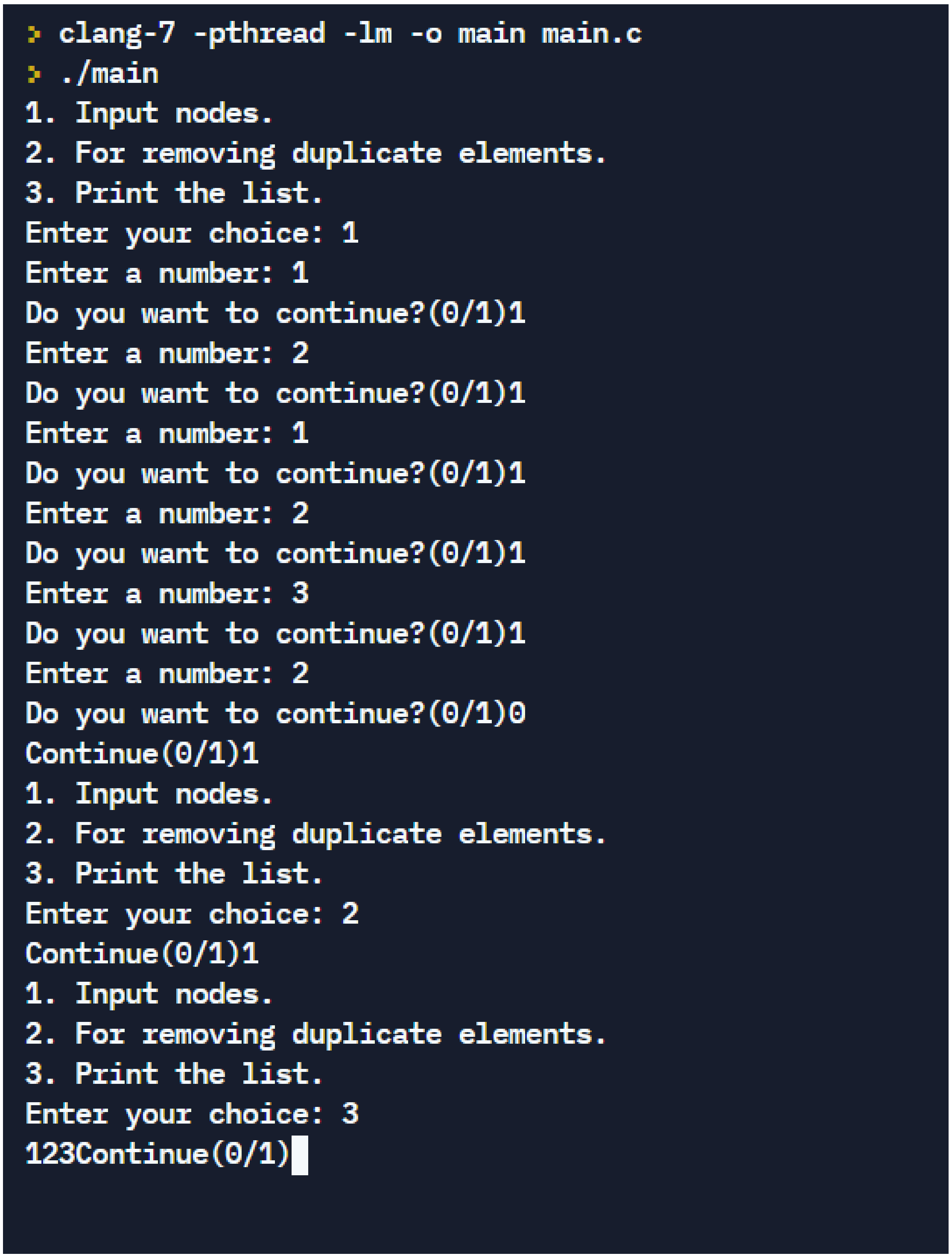
```c
//WAP to remove the duplicates in a sorted double linked list.

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

struct node{
    struct node *prev;
    int num;
    struct node *next;
};
struct node *head=NULL;

void insert()
{
    int n;
    struct node *temp, *newnode;
    while(n)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter a number: ");
        scanf("%d", &newnode->num);
        //fflush(stdin);
        if(head==NULL)
        {
            head=temp=newnode;
        }
        else
        {
                temp->next=newnode;
                newnode->prev=temp;
                temp=newnode;
        }
        printf("Do you want to continue?(0/1)");
        scanf("%d", &n);
    }
}

void removeDuplicateNode()
```



```c
void removeDuplicateNode()
{
    struct node *current, *index, *temp;

    if(head == NULL)
    {
        return;
    }
    else {
        for(current = head; current != NULL; current = current->next)
        {
            for(index = current->next; index != NULL; index = index->next)
            {
                if(current->num == index->num)
                {
                    temp = index;
                    index->prev->next = index->next;

                    if(index->next != NULL)
                    {
                        index->next->prev = index->prev;
                    }
                    temp = NULL;
                }
            }
        }
    }
}

void print()
{
    struct node *temp;
    temp=head;
    while (temp!=NULL)
    {
        printf("%d", temp->num);
        temp=temp->next;
    }

}
```

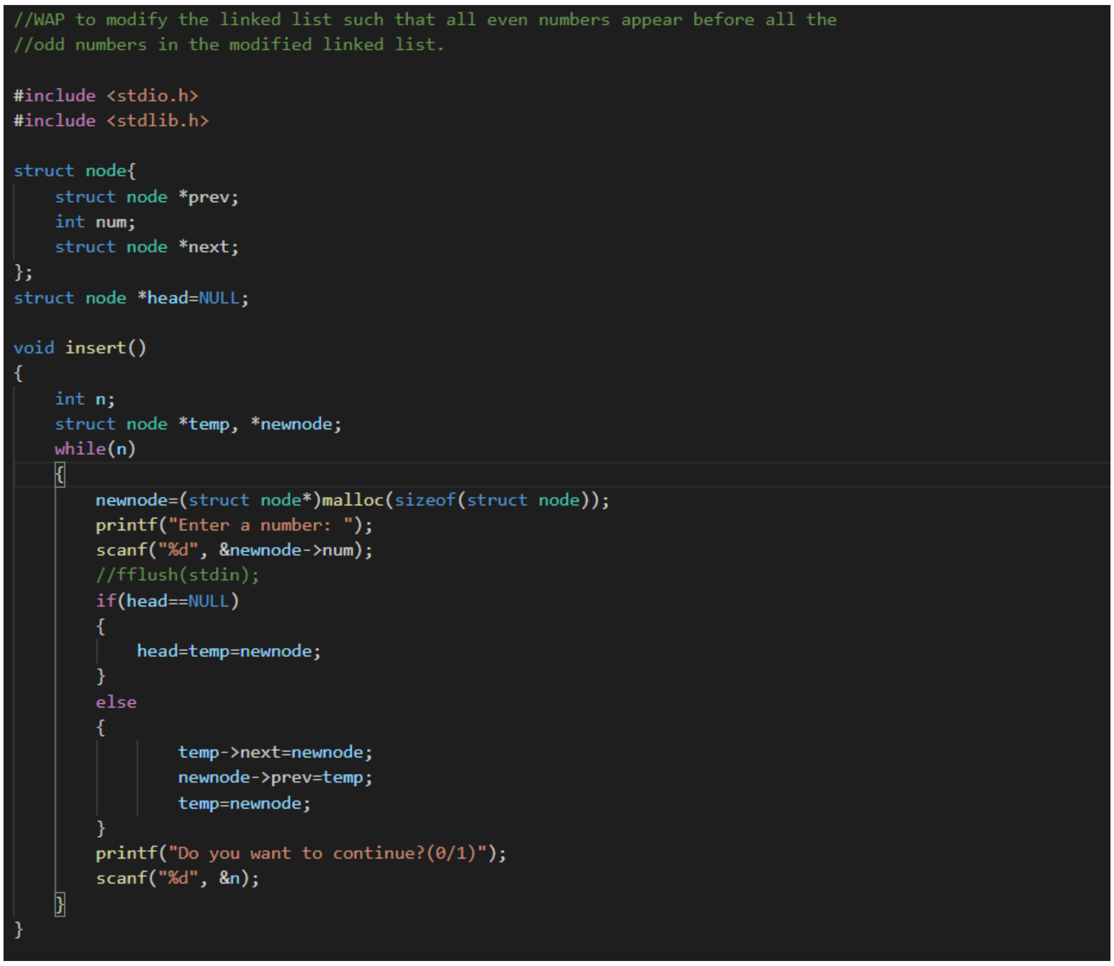


```c
int main()
{
    int ch, end=1;
    while(end)
    {
        printf("1. Input nodes.\n");
        printf("2. For removing duplicate elements.\n");
        printf("3. Print the list.\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                insert();
                break;
            case 2:
                removeDuplicateNode();
                break;
            case 3:
                print();
                break;
            default:
                printf("Invalid input");
        }
        printf("Continue(0/1)");
        scanf("%d", &end);
    }
}
```

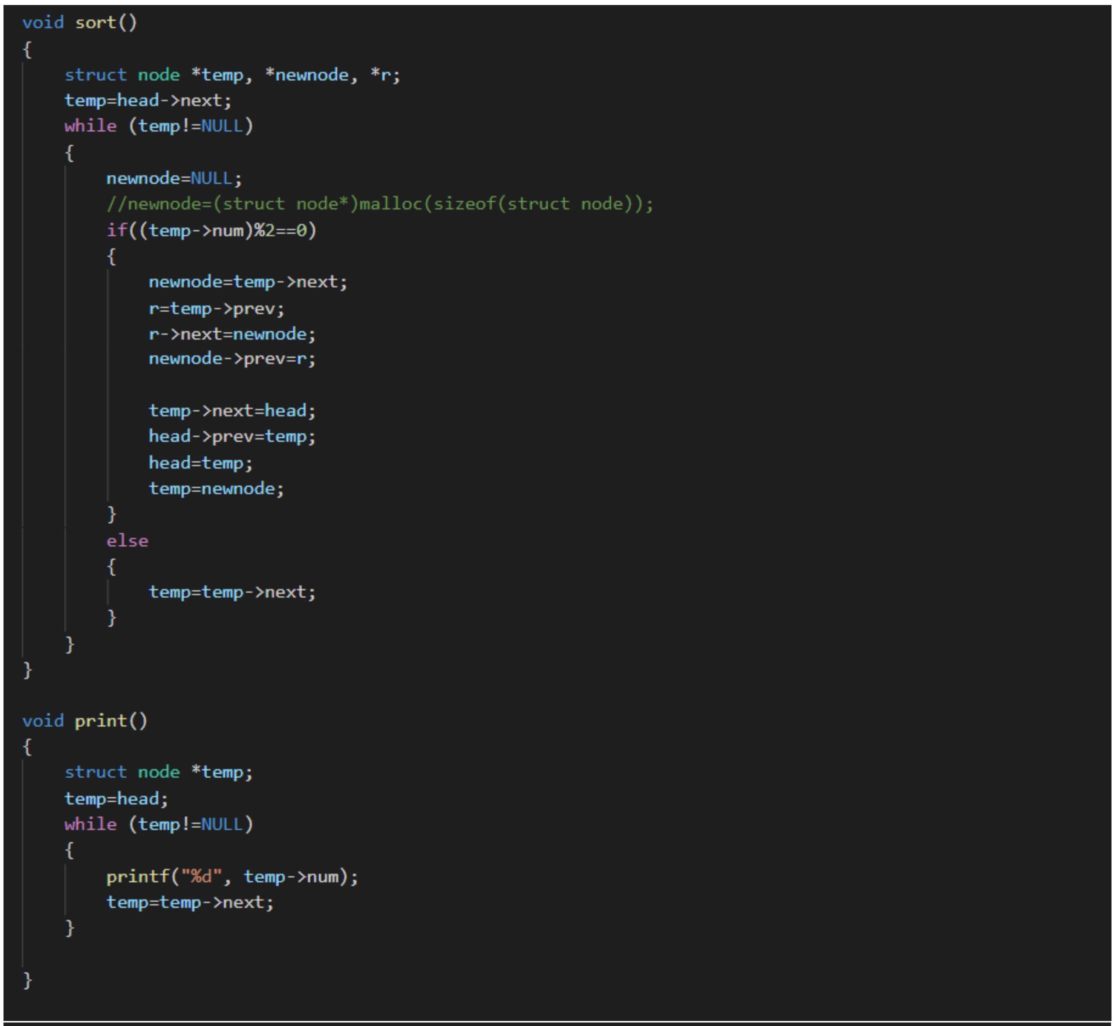


```
> clang-7 -pthread -lm -o main main.c
> ./main
1. Input nodes.
2. For removing duplicate elements.
3. Print the list.
Enter your choice: 1
Enter a number: 1
Do you want to continue?(0/1)1
Enter a number: 2
Do you want to continue?(0/1)1
Enter a number: 1
Do you want to continue?(0/1)1
Enter a number: 2
Do you want to continue?(0/1)1
Enter a number: 3
Do you want to continue?(0/1)1
Enter a number: 2
Do you want to continue?(0/1)0
Continue(0/1)1
1. Input nodes.
2. For removing duplicate elements.
3. Print the list.
Enter your choice: 2
Continue(0/1)1
1. Input nodes.
2. For removing duplicate elements.
3. Print the list.
Enter your choice: 3
123Continue(0/1)
```

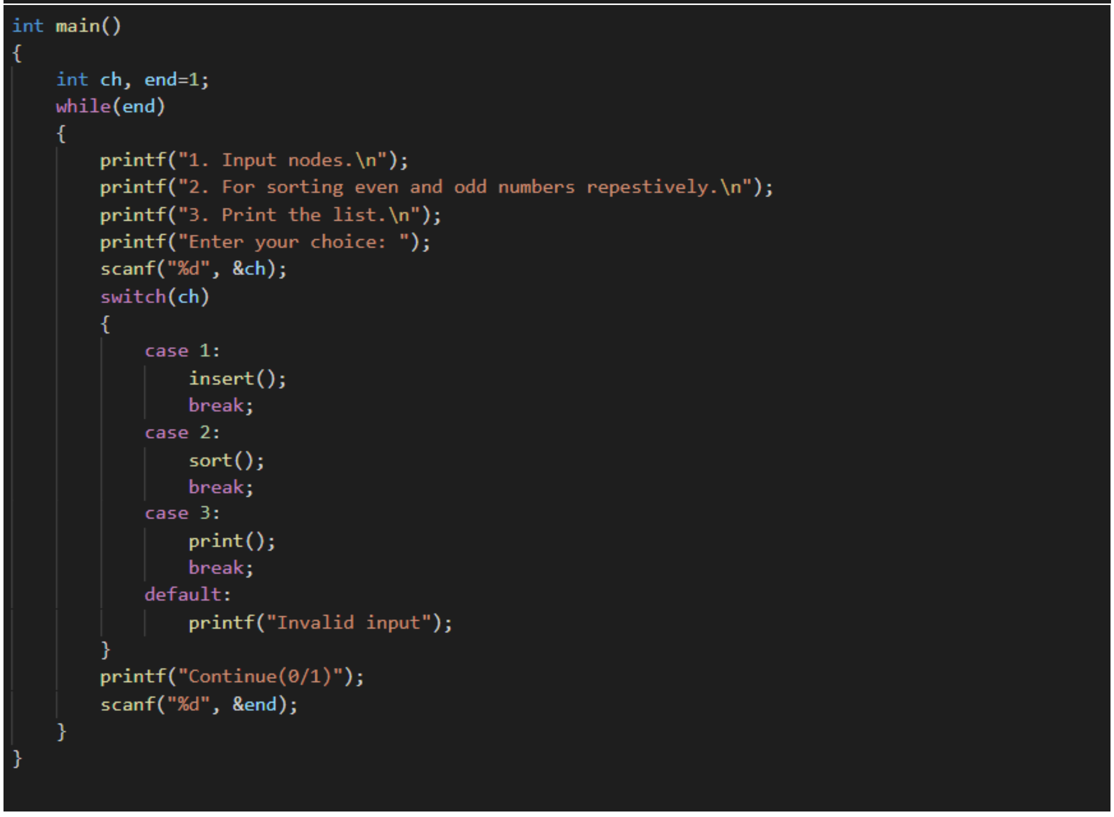
QUESTION 3



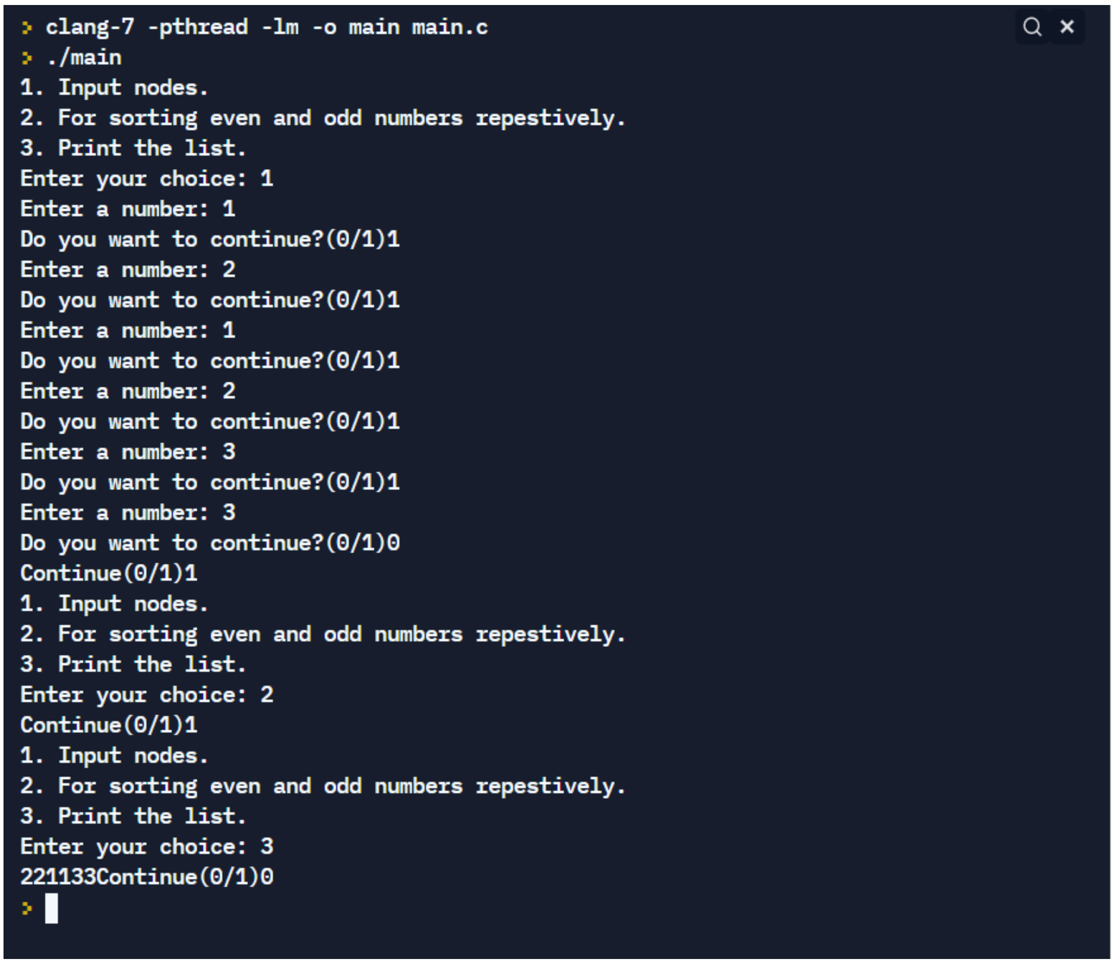
```c
//WAP to modify the linked list such that all even numbers appear before all the
//odd numbers in the modified linked list.

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

struct node{
    struct node *prev;
    int num;
    struct node *next;
};
struct node *head=NULL;

void insert()
{
    int n;
    struct node *temp, *newnode;
    while(n)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter a number: ");
        scanf("%d", &newnode->num);
        //fflush(stdin);
        if(head==NULL)
        {
            head=temp=newnode;
        }
        else
        {
                temp->next=newnode;
                newnode->prev=temp;
                temp=newnode;
        }
        printf("Do you want to continue?(0/1)");
        scanf("%d", &n);
    }
}
```



```c
void sort()
{
    struct node *temp, *newnode, *r;
    temp=head->next;
    while (temp!=NULL)
    {
        newnode=NULL;
        //newnode=(struct node*)malloc(sizeof(struct node));
        if((temp->num)%2==0)
        {
            newnode=temp->next;
            r=temp->prev;
            r->next=newnode;
            newnode->prev=r;

            temp->next=head;
            head->prev=temp;
            head=temp;
            temp=newnode;
        }
        else
        {
            temp=temp->next;
        }
    }
}

void print()
{
    struct node *temp;
    temp=head;
    while (temp!=NULL)
    {
        printf("%d", temp->num);
        temp=temp->next;
    }

}
```

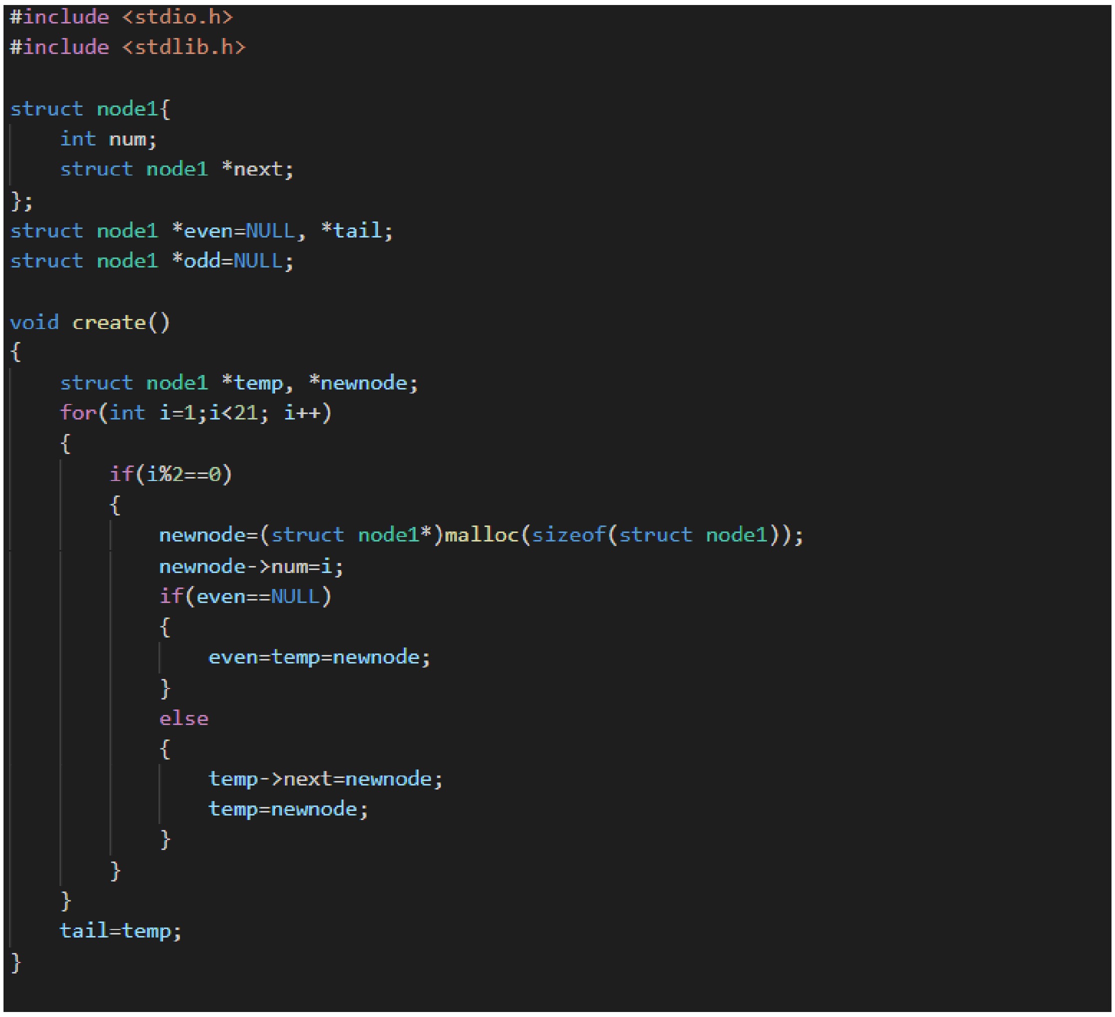


```c
int main()
{
    int ch, end=1;
    while(end)
    {
        printf("1. Input nodes.\n");
        printf("2. For sorting even and odd numbers repestively.\n");
        printf("3. Print the list.\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                insert();
                break;
            case 2:
                sort();
                break;
            case 3:
                print();
                break;
            default:
                printf("Invalid input");
        }
        printf("Continue(0/1)");
        scanf("%d", &end);
    }
}
```

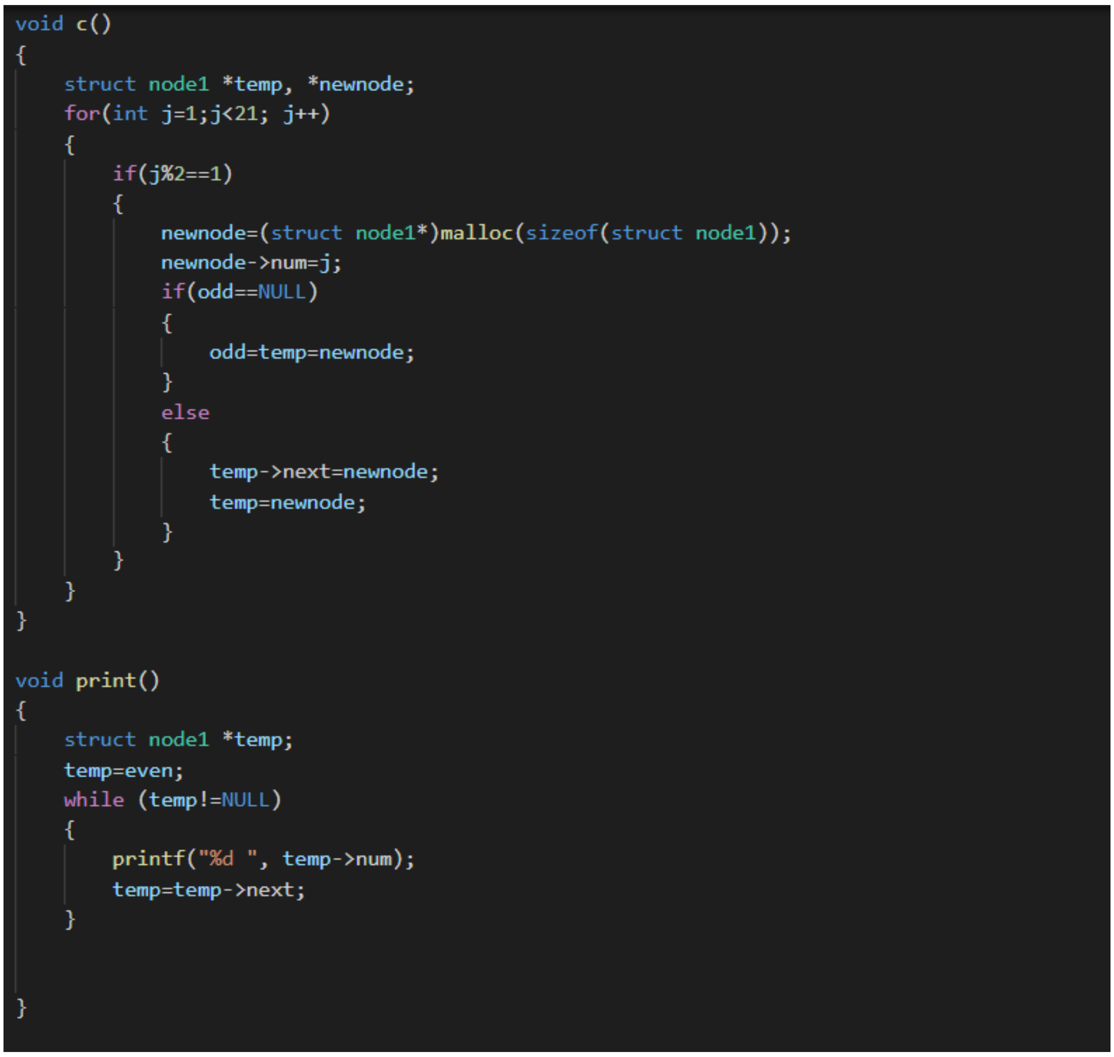


```
> clang-7 -pthread -lm -o main main.c
> ./main
1. Input nodes.
2. For sorting even and odd numbers repestively.
3. Print the list.
Enter your choice: 1
Enter a number: 1
Do you want to continue?(0/1)1
Enter a number: 2
Do you want to continue?(0/1)1
Enter a number: 1
Do you want to continue?(0/1)1
Enter a number: 2
Do you want to continue?(0/1)1
Enter a number: 3
Do you want to continue?(0/1)1
Enter a number: 3
Do you want to continue?(0/1)0
Continue(0/1)1
1. Input nodes.
2. For sorting even and odd numbers repestively.
3. Print the list.
Enter your choice: 2
Continue(0/1)1
1. Input nodes.
2. For sorting even and odd numbers repestively.
3. Print the list.
Enter your choice: 3
221133Continue(0/1)0
> 
```

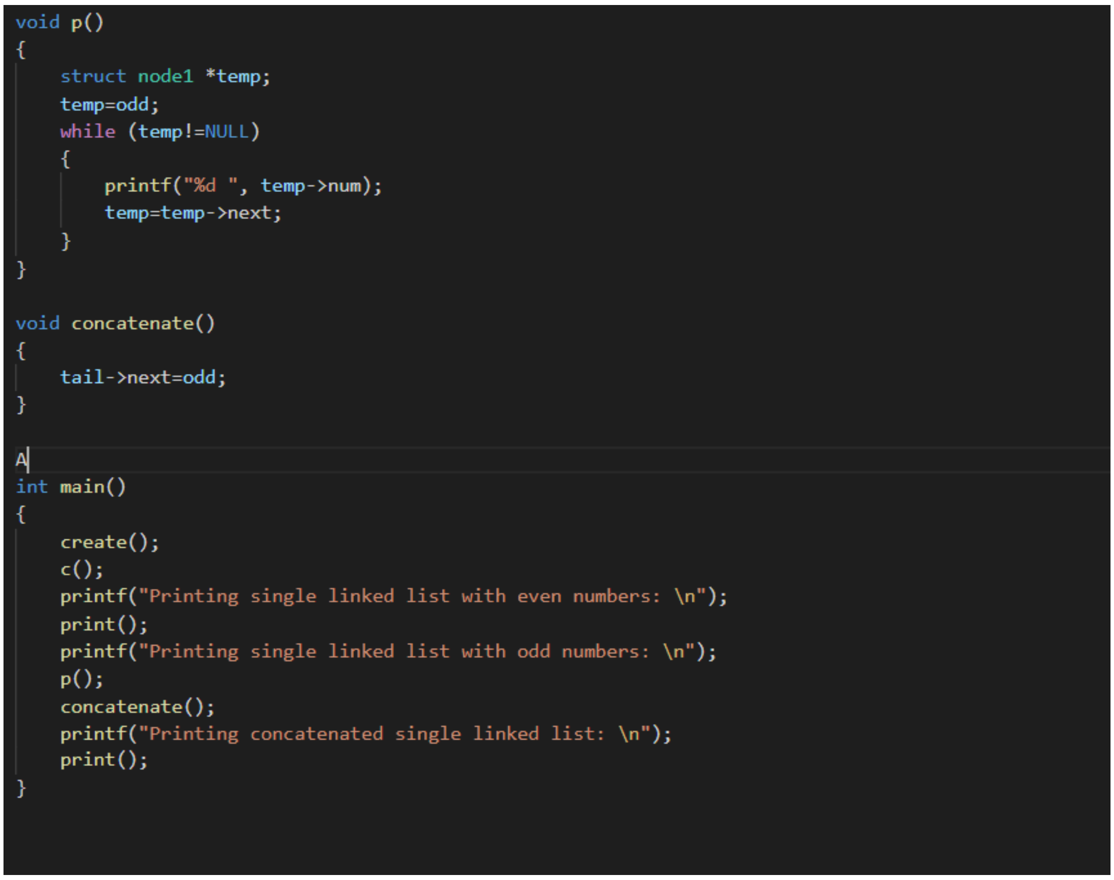
QUESTION 4



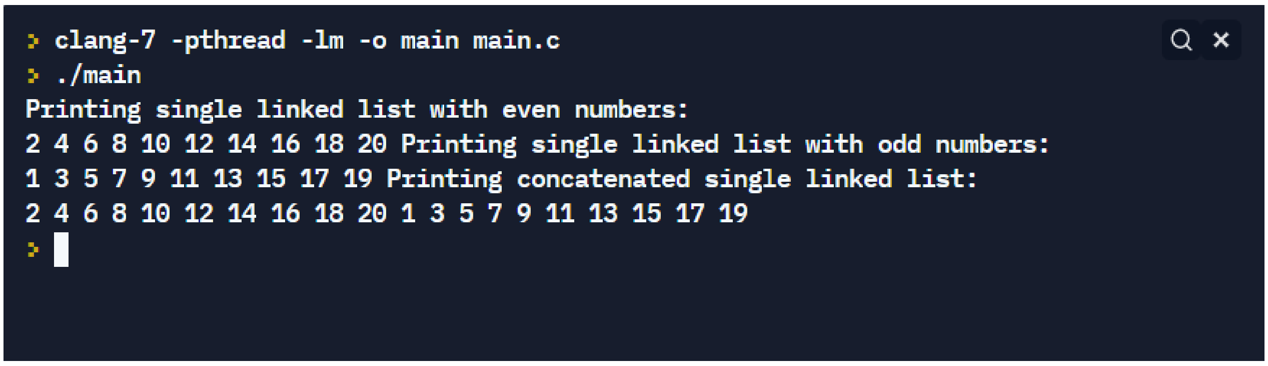
```c
#include <stdio.h>
#include <stdlib.h>

struct node1{
    int num;
    struct node1 *next;
};
struct node1 *even=NULL, *tail;
struct node1 *odd=NULL;

void create()
{
    struct node1 *temp, *newnode;
    for(int i=1;i<21; i++)
    {
        if(i%2==0)
        {
            newnode=(struct node1*)malloc(sizeof(struct node1));
            newnode->num=i;
            if(even==NULL)
            {
                even=temp=newnode;
            }
            else
            {
                temp->next=newnode;
                temp=newnode;
            }
        }
    }
    tail=temp;
}
```



```c
void c()
{
    struct node1 *temp, *newnode;
    for(int j=1;j<21; j++)
    {
        if(j%2==1)
        {
            newnode=(struct node1*)malloc(sizeof(struct node1));
            newnode->num=j;
            if(odd==NULL)
            {
                odd=temp=newnode;
            }
            else
            {
                temp->next=newnode;
                temp=newnode;
            }
        }
    }
}

void print()
{
    struct node1 *temp;
    temp=even;
    while (temp!=NULL)
    {
        printf("%d ", temp->num);
        temp=temp->next;
    }


}
```



```c
void p()
{
    struct node1 *temp;
    temp=odd;
    while (temp!=NULL)
    {
        printf("%d ", temp->num);
        temp=temp->next;
    }
}

void concatenate()
{
    tail->next=odd;
}

A
int main()
{
    create();
    c();
    printf("Printing single linked list with even numbers: \n");
    print();
    printf("Printing single linked list with odd numbers: \n");
    p();
    concatenate();
    printf("Printing concatenated single linked list: \n");
    print();
}
```



```
> clang-7 -pthread -lm -o main main.c
> ./main
Printing single linked list with even numbers:
2 4 6 8 10 12 14 16 18 20 Printing single linked list with odd numbers:
1 3 5 7 9 11 13 15 17 19 Printing concatenated single linked list:
2 4 6 8 10 12 14 16 18 20 1 3 5 7 9 11 13 15 17 19
> 
```