**Q. Write a C program to perform various operations of Doubly Linked List.**

#include <stdio.h>

#include <stdlib.h>

struct student    // creating structure

{

    int roll,age ; char name[30];

    struct student \*next,\*prev;

};

typedef struct student node; // user-defined data type called "node"

node \*create\_list()  // creating linked list

{

    int k,n;

    node \*p,\*head;

    printf("\n How many elements are there in your list? : ");

    scanf("%d",&n);

    for (k=0;k<n;k++)

    {

        if (k==0)

        {

            head=(node\*) malloc(sizeof(node));

            p=head;

        }

        else

        {

            p->next=(node\*) malloc(sizeof(node));

            p=p->next;

        }

        printf("\n Enter details of student in the format [(Roll no.) (Name) (Age)] : ");

        scanf("%d%s%d",&p->roll,p->name,&p->age);

    }

    p->next=NULL;

    return (head);

}

void Display (node\* head)

{

    int count=1;

    node \*p;

    p=head;

    if (head == NULL)  //If there are no details

    {

        printf("\nList is empty");

        return;

    }

    while(p!=NULL)

    {

        printf("\nNode %d - Roll no. : %d, Name : %s, Age : %d",count,p->roll,p->name,p->age);

        count++;

        p=p->next;

    }

}

void Insert(node \*\*head)        // Function to insert at any given position in the linked list

{

    int pos, i=1;

    node \*p, \*ne;

    ne = malloc(sizeof(node));

    ne->next = NULL;

    ne->prev = NULL;

    printf("Enter the position to insert the details at : ");

    scanf("%d", &pos);

    printf("Enter [(Roll no.)(Name)(Age)] : ");

    scanf("%d%s%d",&ne->roll,ne->name,&ne->age);

    p = \*head;

    if (head == NULL)

    {

        \*head = ne;

        ne->prev = NULL;

        ne->next = NULL;

    }

    else if (pos == 1)

    {

        ne->next = \*head;

        ne->next->prev = ne;

        ne->prev = NULL;

        \*head = ne;

    }

    else

    {

        while (i < pos - 1)

        {

            p = p->next;

            i++;

        }

        ne->next = p->next;

        ne->prev = p;

        p->next = ne;

        p->next->prev = ne;

    }

}

void Delete(node \*\*head)        //Function to Delete from any given position of the linked list

{

    int pos, i = 1;

    node \*p, \*position;

    p = \*head;

    if (head == NULL)

        printf("\nList is empty\n");

    else

    {

        printf("\nEnter the position to Delete the details from : ");

        scanf("%d", &pos);

        if (pos == 1)

        {

            position = p;

            p = p->next;

            if (p != NULL)

            {

                p->prev = NULL;

            }

            free(position);

            return;

        }

        while (i < pos - 1)

        {

            p = p->next;

            i++;

        }

        position = p->next;

        if (position->next != NULL)

        {

            position->next->prev = p;

            p->next = position->next;

        }

        free(position);

    }

}

int main()

{

node \*head;

head=create\_list();

Display(head);

int choice;

while(1)

  {

    printf("\n\nMENU : 1-Insert, 2-Delete, 3-Exit\n");

    printf("Enter your choice : ");

    scanf("%d",&choice);

    switch(choice)

    {

        case 1 : Insert(&head); Display(head);

        break;

        case 2 : Delete(&head); Display(head);

        break;

        case 3 : exit(1);

        break;

        default :

        printf("Incorrect Choice. Try Again \n");

        continue;

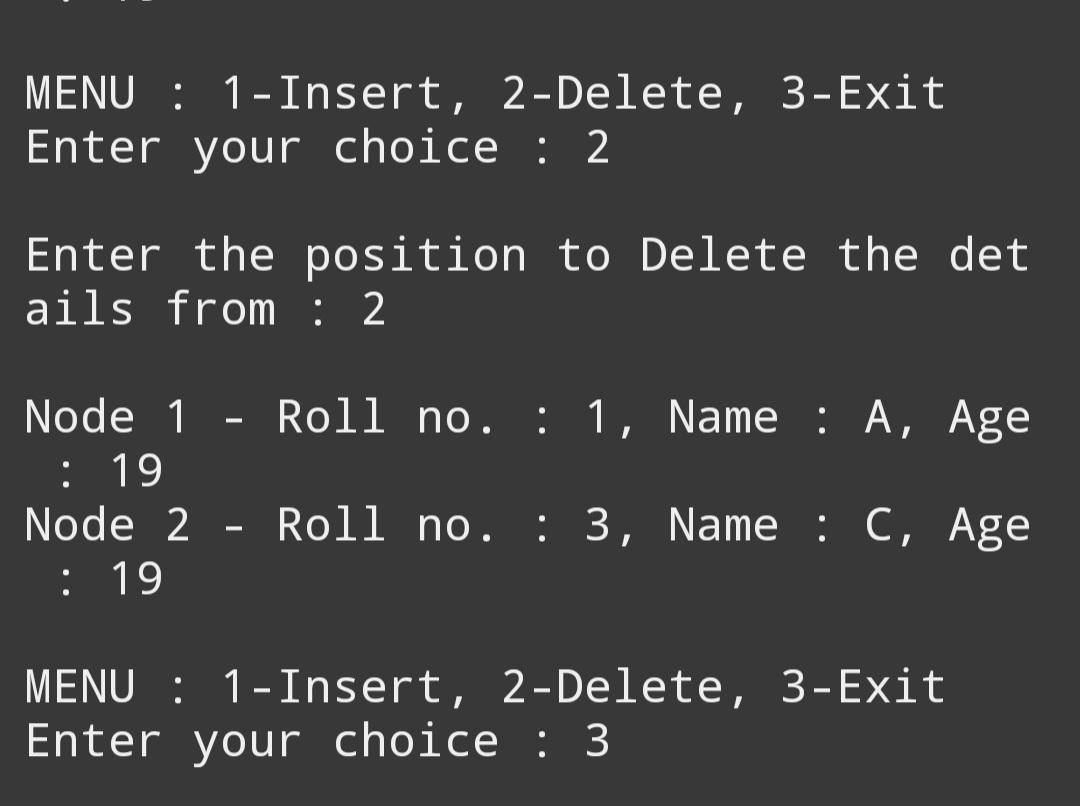
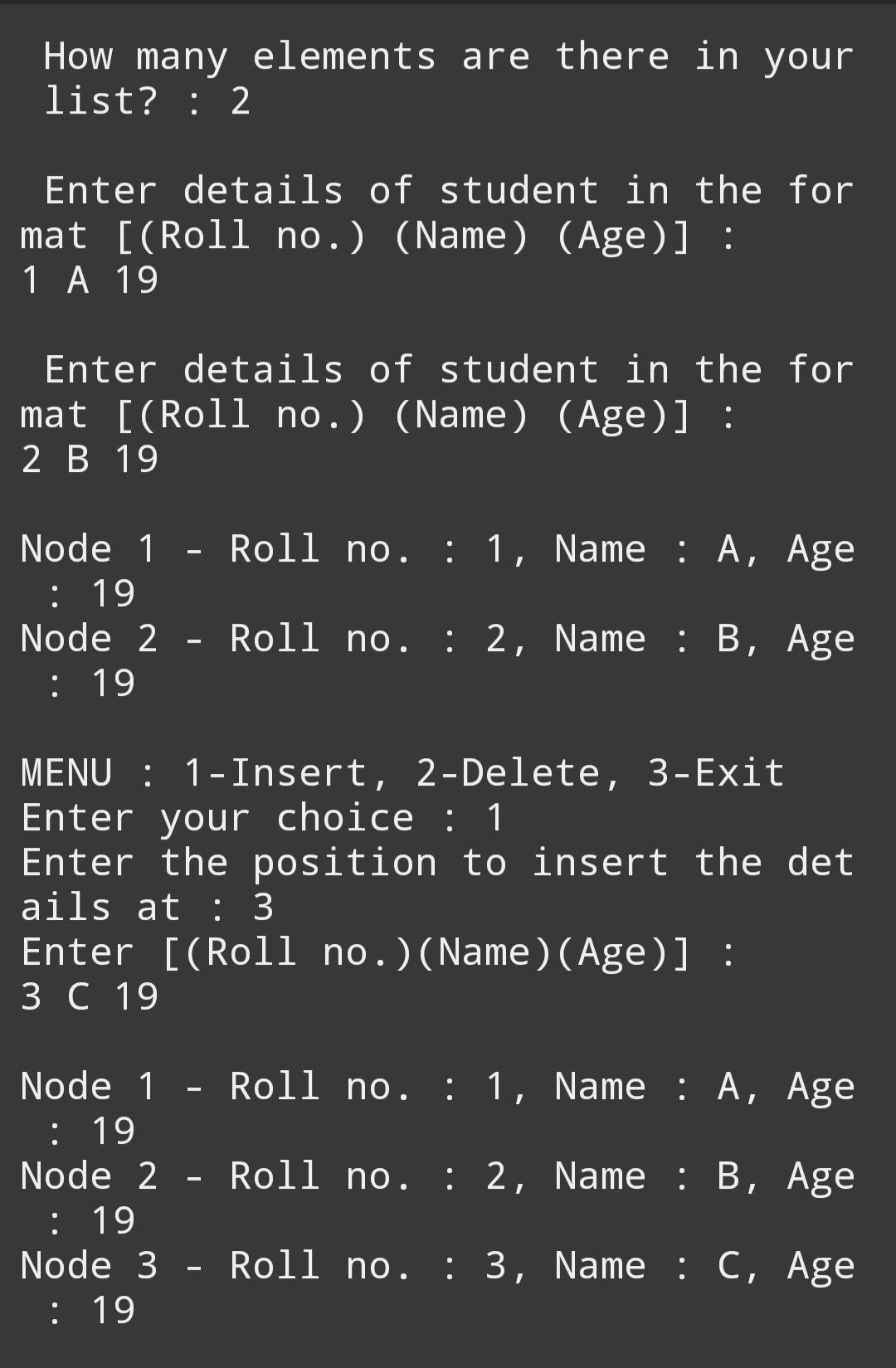
    }

  }

   return 0;

}

**OUTPUT :**

****