**1.**

#include <stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node{

    int data;

    struct node \*next;

};

struct node \* head = NULL;

int countElem(struct node \*head){

    struct node \* ptr = head;

    int count = 0;

    while(ptr!=NULL){

        printf("%d\t",ptr->data);

        ptr = ptr->next;

        count++;

    }

    printf("\n");

    return count;

}

struct node \* insert\_begin(struct node \* head, struct node \* newnode  ){

    if(head == NULL){

        head = newnode;

    }

    else

    {

        newnode->next = head;

        head = newnode;

    }

    return head;

}

struct node \*insert\_end(struct node \* head, struct node \* newnode){

    struct node \* ptr = head;

    while (ptr->next != NULL)

    {

       ptr = ptr->next;

    }

    ptr->next = newnode;

    return head;

}

struct node \*insert\_any(struct node \* head, int data , int pos, int noofelements){

    struct node \*ptr = head , \*prev = head;

    struct node \*newnode;

    newnode = (struct node \*) malloc(sizeof(struct node));

    newnode->data = data;

    printf("%d ", newnode->data);

    newnode->next = NULL;

    int count = 0;

    if(pos == 1){

        head = insert\_begin(head,newnode);

    }

    else if(noofelements == pos-1){

        head = insert\_end(head,newnode);

    }

    else

    {

       while(ptr->next != NULL){

            if(count == pos-1){

                break;

            }

            prev = ptr;

            ptr = ptr->next;

            count++;

        }

        newnode->next = ptr;

        prev->next = newnode;

    }

    return head;

}

int main(void) {

    int noofelements ;

    int data , pos;

    int stop = 1;

    while(stop){

        noofelements = countElem(head);

        if(noofelements == 0){

            printf("List is empty \n");

            printf("Enter data\n ");

            scanf("%d",&data);

            pos=1;

        }

        else{

            printf("Enter data & position to insert\n ");

            scanf("%d%d",&data , &pos);

        }

        head = insert\_any(head,data,pos,noofelements);

        printf("AFTER INSERTING AT POS %d\n",pos);

        countElem(head);

        printf("PRESS \n1 TO CONTINUE\n0 TO STOP\n");

        scanf("%d",&stop);

    }

    return 0;

}

**2.**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node

{

    int data;

    struct node \* next;

};

struct node \*head = NULL;

struct node \* create(struct node \*head){

    struct node \* ptr , \*newnode;

    int data;

    printf("Enter data -1 to exit\n");

    scanf("%d",&data);

    while(data != -1){

        newnode = (struct node \*)malloc(sizeof(struct node));

        newnode->data = data;

        newnode->next = NULL;

        if(head == NULL){

            head=newnode;

        }

        else

        {

            ptr=head;

            while (ptr->next != NULL)

            {

                ptr = ptr->next;

            }

            ptr->next=newnode;

        }

        printf("Enter data -1 to exit\n");

        scanf("%d",&data);

    }

    return head;

}

void print(struct node \*head){

    struct node \* ptr = head;

    while(ptr){

        printf("%d\t",ptr->data);

        ptr=ptr->next;

    }

    printf("\n");

}

struct node \*delete\_beg(struct node \* head){

    int option;

    printf("DO YOU WANT DELETE FIRST ELEMENT\n1 - YES\n0 - NO\n");

    scanf("%d",&option);

    while(option){

        if(head == NULL){

            printf("NO ELEMENTS TO DELETE\n");

            break;

        }

        struct node \* temp = head;

        head=head->next;

        free(temp);

        print(head);

        printf("DO YOU WANT DELETE FIRST ELEMENT\n1 - YES\n0 - NO\n");

        scanf("%d",&option);

    }

    return head;

}

int main(){

    head = create(head);

    print(head);

    head = delete\_beg(head);

    return 0;

}

**3.**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node

{

    int data;

    struct node \* next;

};

struct node \*head = NULL;

struct node \* create(struct node \*head){

    struct node \* ptr , \*newnode;

    int data;

    printf("Enter data -1 to exit\n");

    scanf("%d",&data);

    while(data != -1){

        newnode = (struct node \*)malloc(sizeof(struct node));

        newnode->data = data;

        newnode->next = NULL;

        if(head == NULL){

            head=newnode;

        }

        else

        {

            ptr=head;

            while (ptr->next != NULL)

            {

                ptr = ptr->next;

            }

            ptr->next=newnode;

        }

        printf("Enter data -1 to exit\n");

        scanf("%d",&data);

    }

    return head;

}

void print(struct node \*head){

    struct node \* ptr = head;

    while(ptr){

        printf("%d\t",ptr->data);

        ptr=ptr->next;

    }

    printf("\n");

}

struct node \*delete\_end(struct node \* head){

    int option;

    printf("DO YOU WANT DELETE END ELEMENT\n1 - YES\n0 - NO\n");

    scanf("%d",&option);

    while(option){

        if(head == NULL){

            printf("NO ELEMENTS TO DELETE\n");

            break;

        }

        else if(head != NULL && head->next == NULL){

            head = NULL;

        }

        else

        {

            struct node \* ptr = head , \*prev , \*temp;

            while (ptr->next != NULL)

            {

                prev = ptr;

                ptr=ptr->next;

            }

            temp = prev->next ;

            prev->next = NULL;

            free(temp);

        }

        print(head);

        printf("DO YOU WANT DELETE END ELEMENT\n1 - YES\n0 - NO\n");

        scanf("%d",&option);

    }

    return head;

}

int main(){

    head = create(head);

    print(head);

    head = delete\_end(head);

    return 0;

}

**4.**

#include<stdio.h>

int binary\_search(int arr[] , int beg , int end , int x){

    while (beg < end)

    {

        int mid = beg + (end - beg) / 2;

        if(arr[mid] == x){

            return mid;

        }

        else if(arr[mid] > x)

            end = mid-1;

        else

            beg = mid+1;

    }

    return -1;

}

int main(){

    int arr [] = {2 , 3 , 4 , 10 , 15 , 16 , 17};

    int n = 7, i;

    for(i=0;i<n;i++)

        printf("%d\t",arr[i]);

    printf("\n");

    int x;

    printf("Enter value\n");

    scanf("%d", &x);

    int found = binary\_search(arr , 0 , n-1 , x);

    if(found == -1)

        printf("VALUE IS NOT IN ARRAY\n");

    else

        printf("VALUE FOUND AT POSITION %d\n", (found+1));

    return 0;

}

**5.**

#include<stdio.h>

int ternary\_search(int arr[] , int beg , int end , int x){

    while (beg <= end)

    {

        int mid1 = beg + (end - beg) / 3;

        int mid2 = end - (end - beg) / 3;

        if(arr[mid1] == x)

            return mid1;

        else if(arr[mid2] == x)

            return mid2;

        else if(arr[mid1] > x)

            end = mid1 - 1;

        else if(arr[mid1] < x && arr[mid2] > x){

            beg = mid1 + 1;

            end = mid2 - 1;

        }

        else

            beg = mid2+1;

    }

    return -1;

}

int main(){

    int arr [] = {2 , 3 , 4 , 10 , 15 , 16 , 17};

    int n = 7, i;

    for(i=0;i<n;i++)

        printf("%d\t",arr[i]);

    printf("\n");

    int x;

    printf("Enter value\n");

    scanf("%d", &x);

    int found = ternary\_search(arr , 0 , n-1 , x);

    if(found == -1)

        printf("VALUE IS NOT IN ARRAY\n");

    else

        printf("VALUE FOUND AT POSITION %d\n", (found+1));

    return 0;

}