

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

[Deemed to be University](https://en.wikipedia.org/wiki/Kalinga_Institute_of_Industrial_Technology)

**School of Computer Science and Engineering**

Assignment-II

**13/09/2022**

**Submitted By:-**

**Name:** Abhishikt Mohanty **Roll No:** 21052991

**Section:** CSE-40 **Branch:** Computer Science and Engineering **Subject:** DSA

**Q1. Add two dist (km-m) by passing structure to a function.**

Code:-

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

struct dist

{

    int km;

    int m;

};

void distCal(struct dist d1, struct dist d2)

{

    int dkm, dm;

    dkm = d1.km + d2.km;

    dm = d1.m + d2.m;

    if (dm >= 1000)

    {

        dm -= 1000;

        dkm += 1;

    }

    printf("Distance after adding is %d km & %d m.", dkm, dm);

}

int main()

{

    struct dist d[2];

    int i;

    for (i = 0; i < 2; i++)

    {

        printf("Enter distances %d\nEnter kilo-meter and meter with a space in between them: ", i + 1);

        scanf("%d %d", &d[i].km,&d[i].m);

        printf("\n");

    }

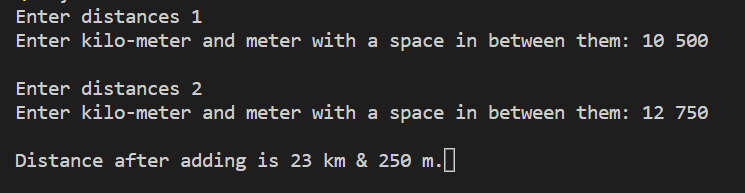
distCal(d[0], d[1]);

    getch();

    return 0;

}

Output:-



**Q2. WAP to print mth node from the last from a list of n nodes.**

Code :-

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int info;

    struct node \*link;

};

struct node \*head;

void create(int n)

{

    struct node \*N, \*M;

    int i = 0;

    while (i < n)

    {

        if (head == NULL)

        {

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

            printf("Enter the info of 1st Node: ");

            scanf("%d", &N->info);

            N->link = NULL;

            head = N;

        }

        else

        {

            printf("Enter info of %d node: ", i + 1);

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

            scanf("%d", &M->info);

            N->link = M;

            M->link = NULL;

            N = M;

        }

        i++;

    }

}

void display()

{

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

    for (temp = head; temp != NULL; temp = temp->link)

        printf("%d ", temp->info);

    printf("\n");

}

void display\_mth(int m, int n)

{

    int i = 1;

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

    for (temp = head; i <= n - m; temp = temp->link, i++)

        ;

    printf("%d ", temp->info);

    printf("\n");

}

int main()

{

    int n, m;

    printf("Enter no of nodes you want to create: ");

    scanf("%d", &n);

    create(n);

    printf("Enter the value of m: ");

    scanf("%d", &m);

    printf("\nList entered is ");

    display();

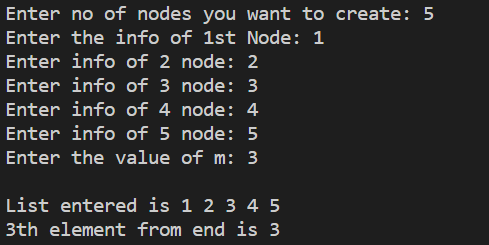
    printf("%dth element from end is ", m);

display\_mth(m, n);

    return 0;

}

Output:-



**Q3. WAP to search an element in a simple linked list, if found delete that node and insert that node at Beginning. Otherwise display an appropriate message.**

Code:-

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int info;

    struct node \*link;

};

struct node \*head;

void create(int n)

{

    struct node \*N, \*M;

    int i = 0;

    while (i < n)

    {

        printf("Enter node %d info: ", i + 1);

        i++;

        if (head == NULL)

        {

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

            scanf("%d", &N->info);

            head = N;

            N->link = NULL;

        }

        else

        {

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

            scanf("%d", &M->info);

            M->link = NULL;

            N->link = M;

            N = M;

        }

    }

}

void display()

{

    struct node \*temp;

    for (temp = head; temp != NULL; temp = temp->link)

        printf("%d ", temp->info);

    printf("\n");

}

void search(int key)

{

    if (head)

    {

        struct node \*ptr, \*temp;

        int flag = 0;

        for (temp = head; temp != NULL; temp = temp->link)

            if (temp->link->info == key)

            {

                flag = 1;

                break;

            }

        if (flag == 1)

        {

            ptr = temp->link;

            temp->link = temp->link->link;

            free(ptr);

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

            newn->info = key;

            newn->link = head;

            head = newn;

            printf("Element Found.\nThe New Linked List is ");

            display();

        }

        else

        printf("Element cannot be found.");

    }

    else

        printf("List is empty, Element cannot be found.");

}

int main()

{

    int n,k;

    printf("Enter no of nodes you want to create: ");

    scanf("%d",&n);

create(n);

    printf("Enter the key: ");

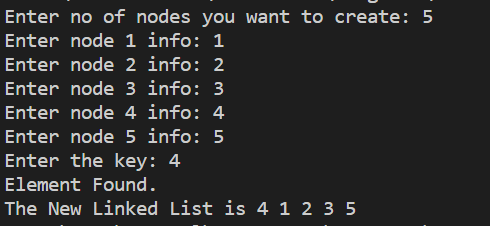
    scanf("%d",&k);

search(k);

    return 0;

}

Output:-



**Q4. WAP to count the no of occurences of an element in a linked list of n nodes.**

Code:-

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int info;

    struct node \*link;

};

struct node \*head;

void create(int n)

{

    struct node \*N, \*M;

    int i = 0;

    while (i < n)

    {

        if (head == NULL)

        {

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

            printf("Enter the info of 1st Node: ");

            scanf("%d", &N->info);

            N->link = NULL;

            head = N;

        }

        else

        {

            printf("Enter info of %d node: ", i + 1);

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

            scanf("%d", &M->info);

            N->link = M;

            M->link = NULL;

            N = M;

        }

        i++;

    }

}

void display()

{

    struct node \*temp;

    for (temp = head; temp != NULL; temp = temp->link)

        printf("%d ", temp->info);

    printf("\n");

}

void ocur(int n)

{

    struct node \*temp;

    int count = 0;

    for (temp = head; temp != NULL; temp = temp->link)

        if(temp->info == n)

        count++;

    printf("No of occurences of %d is %d." ,n,count);

}

int main()

{

    int n,key;

    printf("Enter no of nodes you want to create: ");

    scanf("%d", &n);

    create(n);

    printf("\nList entered is ");

    display();

    printf("\nEnter no to find occurence: ");

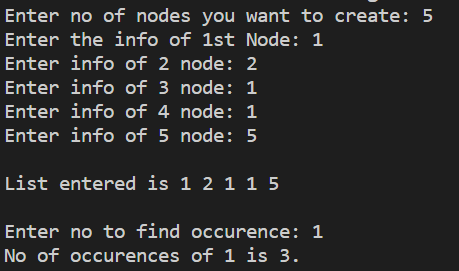
    scanf("%d",&key);

ocur(key);

    return 0;

}

Output:-



**Q5. WAP to check whether a single linked list is palindrome or not.**

Code:-

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int info;

    struct node \*link;

};

struct node \*head;

int count = 0;

void create\_a4(int n)

{

    struct node \*N, \*M;

    int i = 0;

    while (i < n)

    {

        if (head == NULL)

        {

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

            printf("Enter the info of 1st Node: ");

            scanf("%d", &N->info);

            N->link = NULL;

            head = N;

        }

        else

        {

            printf("Enter info of %d node: ", i + 1);

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

            scanf("%d", &M->info);

            N->link = M;

            M->link = NULL;

            N = M;

        }

        i++;

    }

}

void display()

{

    struct node \*temp;

    for (temp = head; temp != NULL; temp = temp->link)

            printf("%d ", temp->info);

    printf("\n");

}

int palindromeCheck (struct node \*p, int count)

{

    int i = 0, j;

    struct node \*front, \*rear;

    while (i != count / 2)

    {

        front = rear = p;

        for (j = 0; j < i; j++)

        {

            front = front->link;

        }

        for (j = 0; j < count - (i + 1); j++)

        {

            rear = rear->link;

        }

        if (front->info != rear->info)

        {

            return 0;

        }

        else

        {

            i++;

        }

    }

    return 1;

}

int main()

{

    int n, key,res;

    printf("Enter no of nodes you want to create: ");

    scanf("%d", &n);

    create(n);

    printf("\nList entered is ");

    display();

    res = palindromeCheck(head,n);

    if(res)

    printf("Palindrome List.");

    else

    printf("Not Palindrome List");

    return 0;

}

Output:-

