6. Design, Develop and Implement a menu driven Program in C for the following operations onCircular QUEUE of Characters (Array Implementation of Queue with maximum size MAX):

a) Insert an Element on to Circular QUEUE.
b) Delete an Element from Circular QUEUE.
c) Demonstrate Overflow and Underflow situations on Circular QUEUE.
d) Display the status of Circular QUEUE.
e) Exit

Support the program with appropriate functions for each of the above operations.

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

#define MAX 5

char circular_queue[MAX];
int front = -1, rear = -1;

int isEmpty()
{
    if (front == -1 && rear == -1)
        return 1;
    else
        return 0;
}

int isFull()
{
    if ((rear + 1) % MAX == front)
        return 1;
    else
        return 0;
}

void insertElement(char element)
{
    if (isFull())
    {
        printf("Circular Queue Overflow\n");
        return;
    }
    else if (isEmpty())
    {
        front = rear = 0;
    }
    else
    {
```

```c
        rear = (rear + 1) % MAX;
    }
    circular_queue[rear] = element;
}

void deleteElement()
{
    if (isEmpty())
    {
        printf("Circular Queue Underflow\n");
        return;
    }
    else if (front == rear)
    {
        front = rear = -1;
    }
    else
    {
        front = (front + 1) % MAX;
    }
}

void display()
{
    int i;
    if (isEmpty())
    {
        printf("Circular Queue is empty\n");
        return;
    }
    printf("Circular Queue elements: ");
    i = front;
    do
    {
        printf("%c ", circular_queue[i]);
        i = (i + 1) % MAX;
    }
    while (i != (rear + 1) % MAX);
    printf("\n");
}

int main()
{
    int choice;
    char element;
    do
    {
        printf("\n\n---- Circular Queue Menu ----\n");
        printf("1. Insert an Element\n");
        printf("2. Delete an Element\n");
        printf("3. Display Circular Queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice)
        {
        case 1:
            printf("Enter element to be inserted: ");
            scanf(" %c", &element);
            insertElement(element);
            break;
        case 2:
            deleteElement();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Please enter a valid option.\n");
        }
    }
    while(choice != 4);

    return 0;
}
```

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo.

a) Create a SLL of N Students Data by using front insertion.

b) Display the status of SLL and count the number of nodes in it

c) Perform Insertion / Deletion at End of SLL

d) Perform Insertion / Deletion at Front of SLL(Demonstration of stack)

e) Exit

```
#include<stdio.h>

#include<stdlib.h>

struct node
{
    char usn[25], name[25], branch[25];
    int sem;
    long int phone;
    struct node * link;
};
typedef struct node * NODE;

NODE start = NULL;
int count = 0;

NODE create()
{
    NODE snode;
    snode = (NODE) malloc(sizeof(struct node));

    if (snode == NULL)
    {
        printf("\nMemory is not available");
        exit(1);
    }
    printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
    scanf("%s %s %s %d %ld", snode -> usn, snode -> name, snode -> branch, & snode
-> sem, & snode -> phone);
    snode -> link = NULL;
    count++;
    return snode;
}

NODE insertfront()
{
    NODE temp;
    temp = create();
    if (start == NULL)
    {
        return temp;
    }
    temp -> link = start;
    return temp;
}

NODE deletefront()
{
    NODE temp;
    if (start == NULL)
    {
        printf("\nLinked list is empty");
        return NULL;
    }

    if (start -> link == NULL)
    {
        printf("\nThe Student node with usn:%s is deleted ", start -> usn);
        count--;
        free(start);
        return NULL;
    }
    temp = start;
    start = start -> link;
    printf("\nThe Student node with usn:%s is deleted", temp -> usn);
    count--;
    free(temp);
    return start;
}

NODE insertend()
{
    NODE cur, temp;
    temp = create();

    if (start == NULL)
    {
        return temp;
    }
    cur = start;
    while (cur -> link != NULL)
    {
        cur = cur -> link;
    }
    cur -> link = temp;
    return start;
}

NODE deleteend()
{
    NODE cur, prev;
    if (start == NULL)
    {
        printf("\nLinked List is empty");
        return NULL;
    }

    if (start -> link == NULL)
    {
        printf("\nThe student node with the usn:%s is deleted", start -> usn);
        free(start);
        count--;
        return NULL;
    }

    prev = NULL;
    cur = start;
    while (cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }

    printf("\nThe student node with the usn:%s is deleted", cur -> usn);
    free(cur);
    prev -> link = NULL;
    count--;
    return start;
}

void display()
{
    NODE cur;
    int num = 1;

    if (start == NULL)
    {

        printf("\nNo Contents to display in SLL \n");
        return;
    }
    printf("\nThe contents of SLL: \n");
    cur = start;
    while (cur != NULL)
    {
        printf("\n|%d| |USN:%s| |Name:%s| |Branch:%s| |Sem:%d| |Ph:%ld|", num, cur
-> usn, cur -> name, cur -> branch, cur -> sem, cur -> phone);
        cur = cur -> link;
        num++;
    }
    printf("\n No of student nodes is %d \n", count);
}

void stackdemo()
{
```

```c
    int ch;
    while (1)
    {
        printf("\n--------Stack Demo using SLL--------\n");
        printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
        printf("\nEnter your choice for stack demo:");
        scanf("%d", & ch);

        switch (ch)
        {
        case 1:
            start = insertfront();
            break;
        case 2:
            start = deletefront();
            break;
        case 3:
            display();
            break;
        default:
            return;
        }
    }
    return;
}

int main()
{
    int ch, i, n;
    while (1)
    {
        printf("\n--------Menu--------");
        printf("\nEnter your choice for SLL operation \n");
        printf("\n1:Create SLL of Student Nodes");
        printf("\n2:DisplayStatus");
        printf("\n3:InsertAtEnd");
        printf("\n4:DeleteAtEnd");
        printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
        printf("\n6:Exit \n");
        printf("\nEnter your choice:");
        scanf("%d", & ch);

        switch (ch)
        {
        case 1:
            printf("\nEnter the no of students: ");
            scanf("%d", & n);
            for (i = 1; i <= n; i++)
                start = insertfront();
            break;

        case 2:
            display();
            break;

        case 3:
            start = insertend();
            break;

        case 4:
            start = deleteend();
            break;

        case 5:
            stackdemo();
            break;

        case 6:
            exit(0);

        default:
            printf("\nPlease enter the valid choice");

        }
    }
}
```

8.Design, Develop and Implement a menu driven Program in C for the following operations onDoubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

a) Create a DLL of N Employees Data by using end insertion.
b) Display the status of DLL and count the number of nodes in it
c) Perform Insertion and Deletion at End of DLL
d) Perform Insertion and Deletion at Front of DLL
e) Demonstrate how this DLL can be used as Double Ended Queue.
f) Exit

```
#include<stdio.h>

#include<stdlib.h>

struct node
{
    char ssn[25], name[25], dept[10], designation[25];
    int sal;
    long int phone;
    struct node * llink;
    struct node * rlink;
};
typedef struct node * NODE;

NODE first = NULL;
int count = 0;

NODE create()
{
    NODE enode;
    enode = (NODE) malloc(sizeof(struct node));
    if (enode == NULL)
    {
        printf("\n Running out of memory ");
        exit(0);
    }
    printf("\n Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the
employee: \n");
    scanf("%s %s %s %s %d %ld", enode -> ssn, enode -> name, enode -> dept, enode -
> designation, & enode -> sal, & enode -> phone);
    enode -> llink = NULL;
    enode -> rlink = NULL;
    count++;
    return enode;
}

NODE insertfront()
{
    NODE temp;
    temp = create();
    if (first == NULL)
    {
        return temp;
    }
    temp -> rlink = first;
    first -> llink = temp;
    return temp;
}

void display()
{
    NODE cur;
    int nodeno = 1;
    cur = first;
    if (cur == NULL)
        printf("\nNo Contents to display in DLL ");
    while (cur != NULL)
    {
        printf("\nENode:%d|SSN:%s| |Name:%s| |Department:%s| |Designation:%s|
|Salary:%d| |Phone no:%ld|", nodeno, cur -> ssn, cur -> name, cur -> dept, cur ->
designation, cur -> sal, cur -> phone);
        cur = cur -> rlink;
        nodeno++;
    }
    printf("\nNo of employee nodes is %d", count);
}

NODE deletefront()
{
    NODE temp;
    if (first == NULL)
    {
        printf("\nDoubly Linked List is empty ");
        return NULL;
    }
    if (first -> rlink == NULL)
    {
        printf("\nThe employee node with the ssn:%s is deleted ", first -> ssn);
        free(first);
        count--;
        return NULL;
    }
    temp = first;
    first = first -> rlink;
    temp -> rlink = NULL;
    first -> llink = NULL;
    printf("\nThe employee node with the ssn:%s is deleted ", temp -> ssn);
    free(temp);
    count--;
    return first;
```

```c
}

NODE insertend()
{
    NODE cur, temp;
    temp = create();

    if (first == NULL)
    {
        return temp;
    }
    cur = first;
    while (cur -> rlink != NULL)
    {
        cur = cur -> rlink;
    }

    cur -> rlink = temp;
    temp -> llink = cur;
    return first;
}

NODE deleteend()
{
    NODE prev, cur;
    if (first == NULL)
    {
        printf("\nDoubly Linked List is empty ");
        return NULL;
    }

    if (first -> rlink == NULL)
    {
        printf("\nThe employee node with the ssn:%s is deleted ", first -> ssn);
        free(first);
        count--;
        return NULL;
    }

    prev = NULL;
    cur = first;

    while (cur -> rlink != NULL)
    {
        prev = cur;
        cur = cur -> rlink;
    }

    cur -> llink = NULL;
    printf("\nThe employee node with the ssn:%s is deleted ", cur -> ssn);
    free(cur);
    prev -> rlink = NULL;
    count--;
    return first;
}

void deqdemo()
{
    int ch;
    while (1)
    {
        printf("\nDemo Double Ended Queue Operation ");
        printf("\n1:InsertQueueFront\n 2: DeleteQueueFront\n 3:InsertQueueRear\n
4:DeleteQueueRear\n 5:DisplayStatus\n 6: Exit \n");
        scanf("%d", & ch);

        switch (ch)
        {
        case 1:
            first = insertfront();
            break;
        case 2:
            first = deletefront();
            break;
        case 3:
            first = insertend();
            break;
        case 4:
            first = deleteend();
            break;
        case 5:
            display();
            break;
        default:
            return;
        }
    }
}

void main()
{
    int ch, i, n;
    while (1)
    {
        printf("\n\n--------Menu--------");
        printf("\n1:Create DLL of Employee Nodes ");
        printf("\n2:DisplayStatus");
        printf("\n3:InsertAtEnd");
        printf("\n4:DeleteAtEnd");
        printf("\n5:InsertAtFront");
        printf("\n6:DeleteAtFront");
        printf("\n7:Double Ended Queue Demo using DLL ");
        printf("\n8:Exit \n");
        printf("\nPlease enter your choice: ");
        scanf("%d", & ch);

        switch (ch)
        {
        case 1:
            printf("\nEnter the no of Employees: ");
            scanf("%d", & n);
            for (i = 1; i <= n; i++)
                first = insertend();
            break;

        case 2:
            display();
            break;

        case 3:
            first = insertend();
            break;

        case 4:
            first = deleteend();
            break;

        case 5:
            first = insertfront();
            break;

        case 6:
            first = deletefront();
            break;

        case 7:
            deqdemo();
            break;

        case 8:
            exit(0);
        default:
            printf("\nPlease Enter the valid choice ");
        }
    }
}
```

Design, Develop and Implement a Program in C for the following operationson SinglyCircular Linked List (SCLL) with header nodes:

a) Represent and Evaluate a Polynomial P(x,y,z) = $6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$.
b) Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z).

Support the program with appropriate functions for each of the above operations.

```c
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define COMPARE(x, y)((x == y) ? 0 : (x > y) ? 1 : -1)

struct node
{
    int coef;
    int xexp, yexp, zexp;
    struct node * link;
};
typedef struct node * NODE;

NODE getnode()
{
    NODE x;
    x = (NODE) malloc(sizeof(struct node));
    if (x == NULL)
    {
        printf("Running out of memory \n");
        return NULL;
    }
    return x;
}

NODE attach(int coef, int xexp, int yexp, int zexp, NODE head)
{
    NODE temp, cur;
    temp = getnode();
    temp -> coef = coef;
    temp -> xexp = xexp;
    temp -> yexp = yexp;
    temp -> zexp = zexp;
    cur = head -> link;
    while (cur -> link != head)
    {
        cur = cur -> link;
    }
    cur -> link = temp;
    temp -> link = head;
    return head;
}

NODE read_poly(NODE head)
{
    int i, j, coef, xexp, yexp, zexp, n;
    printf("\nEnter the no of terms in the polynomial: ");
    scanf("%d", & n);
    for (i = 1; i <= n; i++)
    {
        printf("\n\tEnter the %d term: ", i);
        printf("\n\t\tCoef =  ");
        scanf("%d", & coef);
        printf("\n\t\tEnter Pow(x) Pow(y) and Pow(z): ");
        scanf("%d", & xexp);
        scanf("%d", & yexp);
        scanf("%d", & zexp);
        head = attach(coef, xexp, yexp, zexp, head);
    }
    return head;
}

void display(NODE head)
{
    NODE temp;
    if (head -> link == head)
    {
        printf("\nPolynomial does not exist.");
        return;
    }
    temp = head -> link;

    while (temp != head)
    {
        printf("%dx^%dy^%dz^%d", temp -> coef, temp -> xexp, temp -> yexp, temp ->
zexp);
        temp = temp -> link;
        if (temp != head)
            printf(" + ");
    }
}

int poly_evaluate(NODE head)
{
    int x, y, z, sum = 0;
    NODE poly;

    printf("\nEnter the value of x,y and z: ");
    scanf("%d %d %d", & x, & y, & z);
    poly = head -> link;
    while (poly != head)
    {
        sum += poly -> coef * pow(x, poly -> xexp) * pow(y, poly -> yexp) * pow(z,
poly -> zexp);
        poly = poly -> link;
    }
    return sum;
}

NODE poly_sum(NODE head1, NODE head2, NODE head3)
{
    NODE a, b;
    int coef;
    a = head1 -> link;
    b = head2 -> link;

    while (a != head1 && b != head2)
    {
        while (1)
        {
            if (a -> xexp == b -> xexp && a -> yexp == b -> yexp && a -> zexp == b -> zexp)
            {
                coef = a -> coef + b -> coef;
                head3 = attach(coef, a -> xexp, a -> yexp, a -> zexp, head3);
                a = a -> link;
                b = b -> link;
                break;
            }
            if (a -> xexp != 0 || b -> xexp != 0)
            {
                switch (COMPARE(a -> xexp, b -> xexp))
                {
                case -1:
                    head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
                    b = b -> link;
                    break;

                case 0:
                    if (a -> yexp > b -> yexp)
                    {
                        head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                        a = a -> link;
                        break;
                    }
                    else if (a -> yexp < b -> yexp)
                    {
                        head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
                        b = b -> link;
                        break;
                    }
                    else if (a -> zexp > b -> zexp)
                    {
                        head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                        a = a -> link;
                        break;
                    }
                    else if (a -> zexp < b -> zexp)
                    {
                        head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
                        b = b -> link;
                        break;
                    }
                case 1:
                    head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                    a = a -> link;
                    break;
                }
                break;
            }
            if (a -> yexp != 0 || b -> yexp != 0)
            {
                switch (COMPARE(a -> yexp, b -> yexp))
                {
                case -1:
                    head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
                    b = b -> link;
                    break;
                case 0:
                    if (a -> zexp > b -> zexp)
                    {
                        head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                        a = a -> link;
                        break;
                    }
                    else if (a -> zexp < b -> zexp)
                    {
                        head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
                        b = b -> link;
                        break;
                    }
                case 1:
                    head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                    a = a -> link;
                    break;
                }
                break;
            }
            if (a -> zexp != 0 || b -> zexp != 0)
            {
                switch (COMPARE(a -> zexp, b -> zexp))
                {
                case -1:
                    head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
```

```c
                    b = b -> link;
                    break;
                case 1:
                    head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
                    a = a -> link;
                    break;
                }
                break;
            }
        }
    }
    while (a != head1)
    {
        head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
        a = a -> link;
    }
    while (b != head2)
    {
        head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
        b = b -> link;
    }
    return head3;
}

void main()
{
    NODE head, head1, head2, head3;
    int res, ch;
    head = getnode();
    head1 = getnode();
    head2 = getnode();
    head3 = getnode();

    head -> link = head;
    head1 -> link = head1;
    head2 -> link = head2;
    head3 -> link = head3;

    while (1)
    {
        printf("\n--------Menu--------");
        printf("\n1.Represent and Evaluate a Polynomial P(x,y,z)");
        printf("\n2.Find the sum of two polynomials POLY1(x,y,z)");
        printf("\nEnter your choice:");
        scanf("%d", & ch);
        switch (ch)
        {
        case 1:
            printf("\n----Polynomial evaluation P(x,y,z)----\n");
            head = read_poly(head);
            printf("\nRepresentation of Polynomial for evaluation: \n");
            display(head);
```

```c
                res = poly_evaluate(head);
                printf("\nResult of polynomial evaluation is : %d \n", res);
                break;

            case 2:
                printf("\nEnter the POLY1(x,y,z):  \n");
                head1 = read_poly(head1);
                printf("\nPolynomial 1 is:  \n");
                display(head1);

                printf("\nEnter the POLY2(x,y,z):  \n");
                head2 = read_poly(head2);
                printf("\nPolynomial 2 is: \n");
                display(head2);

                printf("\nPolynomial addition result: \n");
                head3 = poly_sum(head1, head2, head3);
                display(head3);
                break;
            case 3:
                exit(0);
            }
        }
    }
```

10. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers:

a) Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2.
b) Traverse the BST in Inorder, Preorder and Post Order.
c) Search the BST for a given element. (KEY) and report the appropriate message.
d) Exit

```c
#include<stdio.h>

#include<stdlib.h>

struct BST
{
    int data;
    struct BST * lchild;
    struct BST * rchild;
};
typedef struct BST * NODE;

NODE create()
{
    NODE temp;
    temp = (NODE) malloc(sizeof(struct BST));
    printf("\nEnter The value: ");
    scanf("%d", & temp -> data);

    temp -> lchild = NULL;
    temp -> rchild = NULL;
    return temp;
}

void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root);

void insert(NODE root, NODE newnode)
{

    if (newnode -> data < root -> data)
    {
        if (root -> lchild == NULL)
            root -> lchild = newnode;
        else
            insert(root -> lchild, newnode);
    }
    if (newnode -> data > root -> data)
    {
        if (root -> rchild == NULL)
            root -> rchild = newnode;
        else
            insert(root -> rchild, newnode);
    }
}

void search(NODE root)
{
    int key;
    NODE cur;
    if (root == NULL)
    {
        printf("\nBST is empty.");
        return;
    }
    printf("\nEnter Element to be searched: ");
    scanf("%d", & key);
    cur = root;
    while (cur != NULL)
    {
        if (cur -> data == key)
        {
            printf("\nKey element is present in BST ");
            return;
        }
        if (key < cur -> data)
            cur = cur -> lchild;
        else
            cur = cur -> rchild;
    }
    printf("\nKey element is not found in the BST ");
}

void inorder(NODE root)
{
    if (root != NULL)
    {
        inorder(root -> lchild);
        printf("%d ", root -> data);
        inorder(root -> rchild);
    }
}

void preorder(NODE root)
{
    if (root != NULL)
    {
        printf("%d ", root -> data);
        preorder(root -> lchild);
        preorder(root -> rchild);
    }
```

```c
}

void postorder(NODE root)
{
    if (root != NULL)
    {
        postorder(root -> lchild);
        postorder(root -> rchild);
        printf("%d ", root -> data);
    }
}

void main()
{
    int ch, key, val, i, n;
    NODE root = NULL, newnode;
    while (1)
    {
        printf("\n-------BST MENU-------");
        printf("\n1.Create a BST ");
        printf("\n2.Search ");
        printf("\n3.BST Traversals: ");
        printf("\n4.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", & ch);
        switch (ch)
        {
        case 1:
            printf("\nEnter the number of elements: ");
            scanf("%d", & n);
            for (i = 1; i <= n; i++)
            {
                newnode = create();
                if (root == NULL)
                    root = newnode;
                else
                    insert(root, newnode);
            }
            break;
        case 2:
            if (root == NULL)
                printf("\nTree Is Not Created ");
            else
            {
                printf("\nThe Preorder display: ");
                preorder(root);
                printf("\nThe Inorder display: ");
                inorder(root);
                printf("\nThe Postorder display: ");
                postorder(root);
            }
            break;
        case 3:
            search(root);
            break;

        case 4:
            exit(0);
        }
    }
}
```

11. Develop a Program in C for the following operations on Graph(G) of Cities.
a) Create a Graph of N cities using Adjacency Matrix.
b) Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.

```
#include<stdio.h>

#include<stdlib.h>

int a[50][50], n, visited[50];
int q[20], front = -1, rear = -1;
int s[20], top = -1, count = 0;

void bfs(int v)
{
    int i, cur;
    visited[v] = 1;
    q[++rear] = v;
    while (front != rear)
    {
        cur = q[++front];
        for (i = 1; i <= n; i++)
        {
            if ((a[cur][i] == 1) && (visited[i] == 0))
            {
                q[++rear] = i;
                visited[i] = 1;
                printf("%d ", i);
            }
        }
    }
}

void dfs(int v)
{
    int i;
    visited[v] = 1;
    s[++top] = v;
    for (i = 1; i <= n; i++)
    {
        if (a[v][i] == 1 && visited[i] == 0)
        {
            printf("%d ", i);
            dfs(i);
        }
    }
}

int main()
{
    int ch, start, i, j;
    printf("\nEnter the number of vertices in graph:");
    scanf("%d", & n);
    printf("\nEnter the adjacency matrix:\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
            scanf("%d", & a[i][j]);
    }

    for (i = 1; i <= n; i++)
        visited[i] = 0;
    printf("\nEnter the starting vertex: ");
    scanf("%d", & start);

    printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
    printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
    printf("\n==>3:Exit");
    printf("\nEnter your choice: ");
    scanf("%d", & ch);
    switch (ch)
    {
    case 1:
        printf("\nNodes reachable from starting vertex %d are: ", start);
        bfs(start);
        for (i = 1; i <= n; i++)
        {
            if (visited[i] == 0)
                printf("\nThe vertex that is not reachable is %d", i);
        }
        break;

    case 2:
        printf("\nNodes reachable from starting vertex %d are:\n", start);
        dfs(start);
        break;
    case 3:
        exit(0);
    default:
        printf("\nPlease enter valid choice:");
    }
}
```

12. Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let thekeys in K and addresses in L are Integers. Design and develop a Program in C that uses Hashfunction H: K ->L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

```c
#include<stdio.h>

#include<stdlib.h>

int key[20], n, m;
int * ht, index;
int count = 0;

void insert(int key)
{
    index = key % m;
    while (ht[index] != -1)
    {
        index = (index + 1) % m;
    }
    ht[index] = key;
    count++;
}

void display()
{
    int i;
    if (count == 0)
    {
        printf("\nHash Table is empty");
        return;
    }

    printf("\nHash Table contents are:\n ");
    for (i = 0; i < m; i++)
        printf("\n T[%d] --> %d ", i, ht[i]);
}

void main()
{
    int i;
    printf("\nEnter the number of employee  records (N): ");
    scanf("%d", & n);

    printf("\nEnter the two digit memory locations (m) for hash table: ");
    scanf("%d", & m);

    ht = (int * ) malloc(m * sizeof(int));
    for (i = 0; i < m; i++)
        ht[i] = -1;

    printf("\nEnter the four digit key values (K) for N Employee Records:\n ");
    for (i = 0; i < n; i++)
        scanf("%d", & key[i]);

    for (i = 0; i < n; i++)
    {
        if (count == m)
        {
            printf("\n-----Hash table is full. Cannot insert the record %d key-----
", i + 1);
            break;
        }
        insert(key[i]);
    }

    display();
}
```