



IN THE NAME OF ALLAH, THE GREATEST THE MOST MERCIFUL

INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Lab Assignment 4

Matric / ID No. : C211032
ID No. (in words) : C-Two-One-One- Zero-Three-Two
Name : Rashedul Arefin Ifty
Semester : 3rd
Section : 3AM

Course Code : CSE-2322
Course Title : Data Structure Lab
Course Teacher Name : Prof. Mohammed Shamsul Alam

Email ID : r.a.ifty2001@gmail.com
Contact Number : 01840003222



Submitted To:
Prof. Mohammed Shamsul Alam
Professor
Department of Computer Science & Engineering
International Islamic University Chittagong

Ans to the Question Number – 1

```
#include<bits/stdc++.h>
using namespace std;
int arr[201032], n;
void InsertionSort()
{
    int ptr, k, temp;
    arr[0] = -201032;
    for (int k = 2 ; k <= n; k++)
    {
        temp = arr[k];
        ptr = k - 1;
        while (temp < arr[ptr])
        {
            arr[ptr + 1] = arr[ptr];
            ptr--;
        }
        arr[ptr + 1] = temp;
    }
}
void show()
{
    cout << endl;
    for (int i = 1; i <= n; i++) cout << arr[i] << " ";
    cout << endl;
}
int main()
{
    cout << "How many Elements: ";
    cin >> n;
    cout << "Enter Elements: ";
    for (int i = 1; i <= n; i++)
    {
        cin >> arr[i];
    }
}
```

```
InsertionSort();  
show();  
return 0;  
}
```

Ans to the Question Number – 2

```
#include<bits/stdc++.h>  
int arr[201032], n;  
using namespace std;  
void SelectionSort(int *array, int size)  
{  
    int i, j, imin;  
    for(i = 0; i<size-1; i++)  
    {  
        imin = i;  
        for(j = i+1; j<size; j++) if(array[j] < array[imin]) imin = j;  
        int temp;  
        temp = array[i];  
        array[i] = array[imin];  
        array[imin] = temp;  
    }  
}  
int main()  
{  
    cout << "Enter How many elements: ";  
    cin >> n;  
    cout << "Enter your elements: ";  
    for(int i= 0; i<n; i++) cin >> arr[i];  
    cout << endl;  
    cout << "Array is Sorted & sorted elements are: ";  
    SelectionSort(arr, n);  
    for(int i= 0; i<n; i++) cout << arr[i] << " ";  
    cout << endl;  
    return 0;  
}
```

Ans to the Question Number – 3

```
#include<bits/stdc++.h>
using namespace std;
int compareTo(const void* first, const void* second)
{
    int* x = (int*) first;
    int* y = (int*) second;
    if (*x > *y)
    {
        return +1;
    }
    else if (*x < *y)
    {
        return -1;
    }
    else
    {
        return 0;
    }
}
int main()
{
    int arr[201032], n;
    cout << "How many elements?: ";
    cin >> n;
    cout << "Enter the array elements: ";
    for (int i = 0; i < n; i++) cin >> arr[i];
    qsort(arr, n, sizeof(int), compareTo);
    cout << endl << "After sorting: ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    return 0;
}
```

Ans to the Question Number – 4

```
#include<bits/stdc++.h>
using namespace std;
int arr[1000], arr2[1000], arr3[1000000], N, R, S;
void display();
void MergingSort()
{
    int NA = 1, NB = 1, Ptr = 1;
    while (NA <= R && NB <= S)
    {
        if (arr[NA] < arr2[NB])
        {
            arr3[Ptr] = arr[NA];
            Ptr++;
            NA++;
        }
        else
        {
            arr3[Ptr] = arr2[NB];
            Ptr++;
            NB++;
        }
    }
    if (NA > R)
    {
        for (int k = 0; k <= S - NB; k++)
        {
            arr3[Ptr + k] = arr2[NB + k];
        }
    }
    else
    {
        for (int k = 0; k <= R - NA; k++)
        {
            arr3[Ptr + k] = arr[NA + k];
        }
    }
}
```

```

        display();
    }
    void display()
    {
        N = R + S;
        cout << "\nMerged Array Elements: ";
        for (int i = 1; i <= N; i++) cout << arr3[i] << " ";
        cout << endl;
    }

    int main()
    {
        cout << "How Many elements (Array arr) : ";
        cin >> R;
        cout << "Enter sorted elements: ";
        for (int i = 1; i <= R; i++) cin >> arr[i];
        cout << "How Many elements (Array arr2) : ";
        cin >> S;
        cout << "Enter sorted elements : ";
        for (int i = 1; i <= S; i++) cin >> arr2[i];
        MergingSort();
        return 0;
    }

```

Ans to the Question Number – 5

```

#include<bits/stdc++.h>
using namespace std;
int arr[1000], arr2[1000], arr3[1000000], N, R, S;
void MergingSort()
{
    int NA = 1, NB = 1, Ptr = 1;
    while (NA <= R && NB <= S)
    {
        if (arr[NA] < arr2[NB])
        {
            arr3[Ptr] = arr[NA];

```

```

        Ptr++;
        NA++;
    }
    else
    {
        arr3[Ptr] = arr2[NB];
        Ptr++;
        NB++;
    }
}
if (NA > R)
{
    for (int k = 0; k <= S - NB; k++)
    {
        arr3[Ptr + k] = arr2[NB + k];
    }
}
else
{
    for (int k = 0; k <= R - NA; k++)
    {
        arr3[Ptr + k] = arr[NA + k];
    }
}
}
void display()
{
    N = R + S;
    cout << "\nMerged Array Elements : ";
    for (int i = 1; i <= N; i++) cout << arr3[i] << " ";
    cout << endl;
}
int main()
{
    cout << "How Many elements (Array arr): ";
    cin >> R;
    cout << "Enter sorted elements: ";
    for (int i = 1; i <= R; i++) cin >> arr[i];

```

```

cout << "How Many elements (Array arr2): ";
cin >> S;
cout << "Enter sorted elements: ";
for (int i = 1; i <= S; i++) cin >> arr2[i];
MergingSort();
display();
return 0;
}

```

Ans to the Question Number – 6

```

#include<bits/stdc++.h>
using namespace std;
struct node
{
    int info;
    struct node *left;
    struct node *right;
};
node *root;
int insertNode(int Item)
{
    node *p, *newNode, *Back;
    p = root;
    Back = NULL;
    newNode = new node();
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->info = Item;
    while (p != NULL)
    {
        Back = p;
        if (p->info > Item) p = p->left;
        else p = p->right;
    }
    if (Back == NULL) root = newNode;
}

```



```

    else if (Back->info > Item) Back->left = newNode;
    else Back->right = newNode;
    return 0;
}
void inOrder(node *p)
{
    if (p != NULL)
    {
        inOrder(p->left);
        printf("%d ", p->info);
        inOrder(p->right);
    }
}
void preOrder(node *p)
{
    if (p != NULL)
    {
        printf("%d ", p->info);
        preOrder(p->left);
        preOrder(p->right);
    }
}
void postOrder(node *p)
{
    if (p != NULL)
    {
        postOrder(p->left);
        postOrder(p->right);
        printf("%d ", p->info);
    }
}
int menu()
{
    int n;
    cout << "\n\nMain Menu\n";
    cout << "1. Insert\n";
    cout << "2. Display\n";
    cout << "3. Exit\n\n";
}

```

```

    cout << "Enter Choice(1-3): ";
    cin >> n;
    cout << "\n";
    return n;
}
void Display()
{
    if (root)
    {
        cout << "\nTraverse Tree INorder\n";
        inOrder(root);
        cout << "\nTraverse Tree PREorder\n";
        preOrder(root);
        cout << "\nTraverse Tree POSTorder\n";
        postOrder(root);
    }
    else cout << "\nBST IS NULL\n";
}
int main()
{
    node p;
    int VAL, n;
    n = menu();
    do
    {
        if (n == 1)
        {
            cout << "\nInsert a val :";
            cin >> VAL;
            insertNode(VAL);
        }
        if (n == 2)
        {
            Display();
        }
        if (n == 3)
        {
            cout << "\n";

```

```

        break;
    }
    if (n > 3) cout << "\nWrong Choice\n";
    n = menu();
}
while (1);
return 0;
}

```

Ans to the Question Number – 7

```

#include<bits/stdc++.h>
using namespace std;
struct nodeType
{
    int info;
    struct nodeType *left;
    struct nodeType *right;
};
typedef struct nodeType *nodeptr;
nodeptr root;
nodeptr loc, par, save;
int insertNode(int Item)
{
    nodeptr p, newNode, back;
    p = root;
    back = NULL;
    newNode = (nodeType *) malloc(sizeof(nodeType));
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->info = Item;
    while (p != NULL)
    {
        back = p;
        if (p->info > Item) p = p->left;
    }
}

```

```

        else p = p->right;
    }
    if (back == NULL) root = newNode;
    else if (back->info > Item) back->left = newNode;
    else back->right = newNode;
    return 0;
}
void inOrder(nodeptr p)
{
    if (p != NULL)
    {
        inOrder(p->left);
        printf("%d ", p->info);
        inOrder(p->right);
    }
}
void preOrder(nodeptr p)
{
    if (p != NULL)
    {
        printf("%d ", p->info);
        preOrder(p->left);
        preOrder(p->right);
    }
}
void postOrder(nodeptr p)
{
    if (p != NULL)
    {
        postOrder(p->left);
        postOrder(p->right);
        printf("%d ", p->info);
    }
}
int menu()
{
    int n;
    printf("\n\nMain Menu\n");

```

```

    printf("1. Insert\n");
    printf("2. Display\n");
    printf("3. Exit\n\n");
    cout << "4. search" << endl;
    printf("Enter Choice(1-4): ");
    scanf("%d", &n);
    printf("\n");
    return n;
}
void Search(int item)
{
    nodeType *ptr;
    if (root == NULL)
    {
        loc = NULL;
        par = NULL;
        cout << "\nTree is Empty ! \n";
        return;
    }
    if (item == root->info)
    {
        loc = root;
        par = NULL;
        cout << endl << item << " is Found at Root." << endl;
        return;
    }
    if (item < root->info)
    {
        ptr = root->left;
        save = root;
    }
    else
    {
        ptr = root->right;
        save = root;
    }
    while (ptr != NULL)
    {

```

```

        if (item == ptr->info)
        {
            loc = ptr;
            par = save;
            cout << endl << item << " is Found at location: " << loc << "!
Search is Successful!\nChild of Parent: " << par->info << '!' << endl;
            return;
        }
        if (item < ptr->info)
        {
            save = ptr;
            ptr = ptr->left;
        }
        else
        {
            save = ptr;
            ptr = ptr->right;
        }
    }
    loc = NULL;
    par = save;
    if (loc == NULL) cout << endl << "Opps, " << item << " is not
Found! Search Unsuccessful!!" << endl;
    return;
}

void Display()
{
    if (root)
    {
        printf("\nTraverse Tree INorder\n");
        inOrder(root);
        printf("\nTraverse Tree PREorder\n");
        preOrder(root);
        printf("\nTraverse Tree POSTorder\n");
        postOrder(root);
    }
    else printf("\nBST IS NULL\n");
}

```

```

int main()
{
    nodeptr p;
    int VAL;
    root = NULL;
    char ch[11];
    int n = 2;
    n = menu();
    do
    {
        if (n == 1)
        {
            printf("\nInsert a val :");
            scanf("%d", &VAL);
            insertNode(VAL);
        }
        if (n == 2)
        {
            Display();
        }
        if (n == 3)
        {
            printf("\n");
            break;
        }
        if (n == 4)
        {
            cout << "Enter the item to search: ";
            int ok;
            cin >> ok;
            Search(ok);
        }
        if (n > 4) printf("\nWrong Choice\n");
        n = menu();
    }
    while (1);
    return 0;
}

```

Ans to the Question Number – 8

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
    int info;
    struct node *left;
    struct node *right;
};
node *root, *loc, *par, *save, *child;
int insertNode(int Item)
{
    node *p, *newNode, *Back;
    p = root;
    Back = NULL;
    newNode = new node();
    newNode -> left = NULL;
    newNode -> right = NULL;
    newNode -> info = Item;
    while (p != NULL)
    {
        Back = p;
        if (p->info > Item) p = p->left;
        else p = p->right;
    }
    if (Back == NULL) root = newNode;
    else if (Back->info > Item) Back->left = newNode;
    else Back->right = newNode;
    return 0;
}
void Find(int item)
{
    node *ptr;
    if (root == NULL)
    {
        loc = NULL;
    }
}
```



```
    par = NULL;
    return;
}
if (item == root->info)
{
    loc = root;
    par = NULL;
    return;
}
if (item < root->info)
{
    ptr = root->left;
    save = root;
}
else
{
    ptr = root->right;
    save = root;
}
while (ptr != NULL)
{
    if (item == ptr->info)
    {
        loc = ptr;
        par = save;
        return;
    }
    if (item < ptr->info)
    {
        save = ptr;
        ptr = ptr->left;
    }
    else
    {
        save = ptr;
        ptr = ptr->right;
    }
}
```

```

    loc = NULL;
    par = save;
    return;
}
void CaseA(node *loc, node *par)
{
    if (loc->left == NULL && loc->right == NULL) child = NULL;
    else if (loc->left != NULL) child = loc->left;
    else child = loc->right;
    if (par != NULL)
    {
        if (loc == par->left) par->left = child;
        else par->right = child;
    }
    else root = child;
    return;
}
void CaseB(node *loc, node *par)
{
    node *ptr, *SUC, *PARSUC;
    ptr = loc->right;
    save = loc;
    while (ptr->left != NULL)
    {
        save = ptr;
        ptr = ptr->left;
    }
    SUC = ptr;
    PARSUC = save;
    CaseA(SUC, PARSUC);
    if (par != NULL)
    {
        if (loc == par->left) par->left = SUC;
        else par->right = SUC;
    }
    else root = SUC;
    SUC->left = loc->left;
    SUC->right = loc->right;
}

```

```

    return;
}
void Delete(int item)
{
    Find(item);
    if (loc == NULL)
    {
        cout << "Item not found";
        return;
    }
    if (loc->right != NULL && loc->left != NULL)
    {
        CaseB(loc, par);
    }
    else
    {
        CaseA(loc, par);
    }
    return;
}
void preOrder(node *p)
{
    if (p != NULL)
    {
        printf("%d ", p->info);
        preOrder(p->left);
        preOrder(p->right);
    }
}
void Display()
{
    if (root)
    {
        cout << "\nTraverse Tree PREorder\n";
        preOrder(root);
    }
    else cout << "\nBST IS NULL\n";
}

```

```

int main()
{
    int Num, element, item;
    cout << "How many elements for BST? ";
    cin >> Num;
    cout << "\nEnter elements: ";
    for (int i = 0; i < Num; i++)
    {
        cin >> element;
        insertNode(element);
    }
    Display();
    cout << "\nEnter an element to Delete : ";
    cin >> item;
    Delete(item);
    cout << "After deleting " << item << " node : \n";
    Display();
    return 0;
}

```

Ans to the Question Number – 9

```

#include<bits/stdc++.h>
using namespace std;
int tree[201032], N;
void Insheap(int tree[], int N, int item)
{
    int ptr, PAR;
    N = N + 1;
    ptr = N;
    while (ptr != 0)
    {
        PAR = ptr / 2;
        if (item <= tree[PAR])
        {
            tree[ptr] = item;
            return;
        }
    }
}

```

```

    }
    tree[ptr] = tree[PAR];
    ptr = PAR;
}
tree[1] = item;
return;
}
void Display()
{
    cout << "Maxheap elements: ";
    for (int i = 1; i <= N; i++) cout << tree[i] << " ";
}
int main()
{
    int element;
    cout << "How many element? :";
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> tree[i];
    for (int j = 1; j < N; j++)
    {
        Insheap(tree, j, tree[j + 1]);
    }
    Display();
    return 0;
}

```

Ans to the Question Number – 10

```

#include<bits/stdc++.h>
using namespace std;
int tree[201032], N;
void Insheap(int tree[], int N, int item)
{
    int ptr, PAR;
    N = N + 1;
    ptr = N;
    while (ptr != 0)

```

```

{
    PAR = float(ptr / 2);
    if (item <= tree[PAR])
    {
        tree[ptr] = item;
        return;
    }
    tree[ptr] = tree[PAR];
    ptr = PAR;
}
tree[1] = item;
return;
}
void Delheap()
{
    int item;
    int ptr, left, right, last;
    item = tree[1];
    last = tree[N];
    N = N - 1;
    ptr = 1;
    left = 2;
    right = 3;
    while (right <= N || left <= N)
    {
        if (last >= tree[left] && last >= tree[right])
        {
            tree[ptr] = last;
            return;
        }
        if (tree[right] <= tree[left])
        {
            tree[ptr] = tree[left];
            ptr = left;
        }
        else
        {
            tree[ptr] = tree[right];

```

```

        ptr = right;
    }
    left = 2 * ptr;
    right = left + 1;
}
if (left == N && last < tree[left])
{
    ptr = left;
}
tree[ptr] = last;
return;
}
void Display()
{
    cout << "Maxheap elements: ";
    for (int i = 1; i <= N; i++) cout << tree[i] << " ";
}
int main()
{
    int element;
    cout << "How many element? :";
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> tree[i];
    for (int j = 1; j < N; j++)
    {
        Insheap(tree, j, tree[j + 1]);
    }
    cout << "After deleting ";
    Delheap();
    Display();
    return 0;
}

```

Ans to the Question Number – 11

```

#include<bits/stdc++.h>
using namespace std;

```

```

int tree[201032], N, Size, Item;
void Insheap(int tree[], int N, int item)
{
    int ptr, PAR;
    N = N + 1;
    ptr = N;
    while (ptr != 0)
    {
        PAR = float(ptr / 2);
        if (item <= tree[PAR])
        {
            tree[ptr] = item;
            return;
        }
        tree[ptr] = tree[PAR];
        ptr = PAR;
    }
    tree[1] = item;
    return;
}
void Delheap()
{
    int ptr, left, right, last;
    Item = tree[1];
    last = tree[N];
    N = N - 1;
    ptr = 1;
    left = 2;
    right = 3;
    while (right <= N || left <= N)
    {
        if (last >= tree[left] && last >= tree[right])
        {
            tree[ptr] = last;
            return;
        }
        if (tree[right] <= tree[left])
        {

```



```

        tree[ptr] = tree[left];
        ptr = left;
    }
    else
    {
        tree[ptr] = tree[right];
        ptr = right;
    }
    left = 2 * ptr;
    right = left + 1;
}
if (left == N && last < tree[left])
{
    ptr = left;
}
tree[ptr] = last;
return;
}
void heapsort()
{
    int val, j;
    for (j = 1; j < N; j++)
    {
        val = tree[j + 1];
        Insheap(tree, j, val);
    }
    while (N > 1)
    {
        Delheap();
        tree[N + 1] = Item;
    }
}
void Display()
{
    for (int i = 1; i <= Size; i++) cout << tree[i] << " ";
}
int main()
{

```

```

cout << "How many Elements : ";
cin >> N;
Size = N;
for (int i = 1; i <= N; i++)
{
    cin >> tree[i];
}
cout << "Elements before Heapsort:" << endl;
Display();
cout << endl;
heapsort();
cout << "After Heap sort:" << endl;
Display();
return 0;
}

```

Ans to the Question Number – 12

```

#include<bits/stdc++.h>
using namespace std;
#define Max 32
int adj[ Max ][ Max ];
int n;
int main()
{
    int max_edges, n, i, j, origin, destin;
    char graph_type;
    cout << "Enter number of nodes : ";
    cin >> n;
    cout << "Enter type of graph, directed or undirected (d/u) : ";
    fflush( stdin );
    cin >> graph_type;
    if ( graph_type == 'u' ) max_edges = n * ( n - 1 ) / 2;
    else max_edges = n * ( n - 1 );
    for ( i = 1; i <= max_edges; i++ )
    {
        cout << "Enter edge " << i << " ( 0 0 to quit ) : ";
    }
}

```

```

    cin >> origin >> destin;
    if ( ( origin == 0 ) && ( destin == 0 ) ) break;
    if ( origin > n || destin > n || origin <= 0 || destin <= 0 )
    {
        cout << "Invalid edge!\n" ;
        i--;
    }
    else
    {
        adj[ origin ][ destin ] = 1;
        if ( graph_type == 'u' ) adj[ destin ][ origin ] = 1;
    }
}
cout << "The adjacency matrix is :\n" ;
for ( i = 1 ; i <= n; i++ )
{
    for ( j = 1; j <= n; j++ ) printf( "%4d", adj[ i ][ j ] );
    cout << "\n" ;
}
return 0;
}

```

Ans to the Question Number – 13

```

#include<stdio.h>
#define MAX 1032
void display(int matrix[MAX][MAX]);
void pow_matrix(int p, int adjp[MAX][MAX] );
void multiply(int mat1[MAX][MAX], int mat2[MAX][MAX], int
mat3[MAX][MAX]);
void create_graph( );
int adj[MAX][MAX];
int n;
void create_graph()
{

```

```

int i, max_edges, origin, destin;
printf("\nEnter number of vertices : ");
scanf("%d", &n);
max_edges = n * (n - 1);
for ( i = 1; i <= max_edges; i++ )
{
    printf("\nEnter edge %d( -1 -1 ) to quit : ", i);
    scanf("%d %d", &origin, &destin);
    if ( (origin == -1) && (destin == -1) ) break;
    if ( origin >= n || destin >= n || origin < 0 || destin < 0 )
    {
        printf("\nInvalid edge!\n");
        i--;
    }
    else adj[origin][destin] = 1;
}
}

void pow_matrix(int p, int adjp[MAX][MAX])
{
    int i, j, k, tmp[MAX][MAX];
    for (i = 0; i < n; i++) for (j = 0; j < n; j++) adjp[i][j] = adj[i][j];
    for (k = 1; k < p; k++)
    {
        multiply(adjp, adj, tmp);
        for (i = 0; i < n; i++) for (j = 0; j < n; j++) adjp[i][j] = tmp[i][j];
    }
}

void multiply(int mat1[MAX][MAX], int mat2[MAX][MAX], int
mat3[MAX][MAX])
{
    int i, j, k;
    for (i = 0; i < n; i++) for (j = 0; j < n; j++)
    {
        mat3[i][j] = 0;
        for (k = 0; k < n; k++) mat3[i][j] = mat3[i][j] + mat1[i][k] *
mat2[k][j];
    }
}

```

```

void display(int matrix[MAX][MAX])
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++) printf("%4d", matrix[i][j]);
        printf("\n");
    }
    printf("\n");
}

int main()
{
    int adjp[MAX][MAX];
    int x[MAX][MAX], path[MAX][MAX], i, j, p;
    create_graph();
    printf("\nThe adjacency matrix is :\n");
    display(adj);
    for (i = 0; i < n; i++) for (j = 0; j < n; j++) x[i][j] = 0;
    for (p = 1; p <= n; p++)
    {
        pow_matrix(p, adjp);
        printf("\nAdjacency matrix raised to power [ %d ] is - \n", p);
        display(adjp);
        for (i = 0; i < n; i++) for (j = 0; j < n; j++) x[i][j] = x[i][j] +
adjp[i][j];
    }
    printf("\nThe matrix x is :\n");
    display(x);
    for (i = 0; i < n; i++) for (j = 0; j < n; j++) if (x[i][j] == 0 ) path[i][j]
= 0;
        else path[i][j] = 1;
    printf("\nThe path matrix is :\n");
    display(path);
    return 0;
}

```

Ans to the Question Number – 14

```
#include<bits/stdc++.h>
using namespace std;
#define Max 32
int adj[ Max ][ Max ];
int P[ Max ][ Max ];
int n;
int main()
{
    int max_edges, n, i, j, origin, destin;
    char graph_type;
    printf( "Enter number of nodes : " );
    scanf( "%d", &n );
    printf( "Enter type of graph, directed or undirected (d/u) : " );
    fflush( stdin );
    getchar();
    scanf( "%c", &graph_type );
    if ( graph_type == 'u' ) max_edges = n * ( n - 1 ) / 2;
    else max_edges = n * ( n - 1 );
    for ( i = 1; i <= max_edges; i++ )
    {
        printf( "Enter edge %d( 0 0 to quit ) : ", i );
        scanf( "%d %d", &origin, &destin );
        if ( ( origin == 0 ) && ( destin == 0 ) ) break;
        if ( origin > n || destin > n || origin <= 0 || destin <= 0 )
        {
            printf( "Invalid edge!\n" );
            i--;
        }
        else
        {
            adj[ origin ][ destin ] = 1;
            if ( graph_type == 'u' ) adj[ destin ][ origin ] = 1;
        }
    }
    for (int i = 1; i <= n; i++)
    {
```

```

    for (int j = 1; j <= n; j++)
    {
        if (adj[i][j] == 0)
        {
            P[i][j] = 0;
        }
        else P[i][j] = 1;
    }
}
for (int k = 1; k <= n; k++)
{
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            P[i][j] = P[i][j] || (P[i][k] && P[k][j]);
        }
    }
}
printf( "The adjacency matrix is :\n" );
for ( i = 1; i <= n; i++ )
{
    for ( j = 1; j <= n; j++ ) printf( "%4d", adj[ i ][ j ] );
    printf( "\n" );
}
printf( "The Path matrix is :\n" );
for ( int i = 1; i <= n; i++ )
{
    for ( int j = 1; j <= n; j++ ) printf( "%4d", P[ i ][ j ] );
    printf( "\n" );
}
return 0;
}

```

Ans to the Question Number – 15

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int V, x, y, n;
    cin >> V >> n;
    vector<int> adj[V];
    for (int i = 0; i < n; i++)
    {
        cin >> x >> y;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }
    for (int d = 0; d < V; d++)
    {
        cout << endl << "Vertex " << d << ":";
        {
            for (auto i : adj[d]) cout << "-> " << i;
            cout << endl;
        }
    }
    return 0;
}
```

Ans to the Question Number – 16

```
#include<bits/stdc++.h>
using namespace std;
#define MAX 100
#define initial 1
#define waiting 2
#define visited 3
int n;
int adj[MAX][MAX];
```



```

int state[MAX];
void create_graph();
void BF_Traversal();
void BFS(int v);
int Queue[MAX], Front = -1, Rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();
void BF_Traversal()
{
    int v;
    for (v = 0; v < n; v++) state[v] = initial;
    cout << "Enter Start Vertex for BFS: \n";
    cin >> v;
    BFS(v);
}
void BFS(int v)
{
    int i;
    insert_queue(v);
    state[v] = waiting;
    while (!isEmpty_queue())
    {
        v = delete_queue( );
        cout << v;
        state[v] = visited;
        for (i = 0; i < n; i++)
        {
            if (adj[v][i] == 1 && state[i] == initial)
            {
                insert_queue(i);
                state[i] = waiting;
            }
        }
    }
    cout << endl;
}
void insert_queue(int vertex)

```

```

{
    if (Rear == MAX - 1) cout << "Queue Overflow\n";
    else
    {
        if (Front == -1) Front = 0;
        Rear = Rear + 1;
        Queue[Rear] = vertex ;
    }
}

int isEmpty_queue()
{
    if (Front == -1 || Front > Rear) return 1;
    else return 0;
}

int delete_queue()
{
    int delete_item;
    if (Front == -1 || Front > Rear)
    {
        cout << "Queue Underflow\n";
        exit(1);
    }
    delete_item = Queue[Front];
    Front = Front + 1;
    return delete_item;
}

void create_graph()
{
    int count, max_edge, origin, destin;
    cout << "Enter number of vertices : ";
    cin >> n;
    max_edge = n * (n - 1);
    for (count = 1; count <= max_edge; count++)
    {
        cout << "Enter edge " << count << "( -1 -1 to quit ) : ";
        cin >> origin >> destin;
        if ((origin == -1) && (destin == -1)) break;
        if (origin >= n || destin >= n || origin < 0 || destin < 0)

```

```

        {
            cout << "Invalid edge!\n";
            count--;
        }
        else
        {
            adj[origin][destin] = 1;
        }
    }
}
int main()
{
    create_graph();
    BF_Traversal();
    return 0;
}

```

Ans to the Question Number – 17

```

#include<bits/stdc++.h>
using namespace std;
int A[100][100], s[100], visited[100], n, i, j, top = 0;
void DFS(int v)
{
    for (i = 1; i <= n; i++)
    {
        if (A[v][i] && !visited[i])
        {
            s[++top] = i;
        }
    }
    if (top != 0)
    {
        visited[s[top]] = 1;
        DFS(s[top--]);
    }
}

```

```
int main()
{
    int v;
    cout << " Enter the number of nodes : ";
    cin >> n;
    cout << " Enter the adjacency matrix : ";
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            cin >> A[i][j];
        }
    }
    cout << " Enter the starting node : ";
    cin >> v;
    for (i = 1; i <= n; i++)
    {
        s[i] = 0;
        visited[i] = 0;
    }
    DFS(v);
    cout << " The reachable nodes are : ";
    for (i = 1; i <= n; i++)
    {
        if (visited[i] != 0)
        {
            cout << endl << " The node " << i << " is reachable " ;
        }
        else
        {
            cout << endl << " The node " << i << " is not reachable " ;
        }
    }
    return 0;
}
```

Ans to the Question Number – 18

```
##include<bits/stdc++.h>
using namespace std;
#define SIZE 10
int H[SIZE + 1];
#define m 7 void Insert()
{
    int key, index, n = 0;
    cout << "Enter key element to insert\n";
    cin >> key;
    index = (key % m) + 1;
    while (H[index] != 0)
    {
        if (H[index] == 0) break;
        index++;
        n++;
        if (index == SIZE + 1) index = 1;
        if (n == SIZE + 1) break;
    }
    if (n == SIZE + 1)
    {
        cout << "\nHash Table is full of elements\nNo Place to insert
this element\n\n";
    }
    else H[index] = key;
}
void Search()
{
    int key, index, n = 0;
    cout << "\nEnter the element you want to search\n";
    cin >> key;
    index = (key % m) + 1;
    while (n != SIZE)
    {
        if (H[index] == key)
        {
            cout << "Element found at index " << index << "\n";
        }
    }
}
```

```

        break;
    }
    else
    {
        if (H[index] == 0)
        {
            cout << "Element not found in Hash table\n";
            break;
        }
        if (H[index] == -1)
        {
            index++;
        }
        n++;
        index++;
        if (index == SIZE) index = 0;
    }
}
if (n-- == SIZE) cout << "Element not found in Hash table\n";
}
void display()
{
    int i;
    cout << "Index\tValue\n";
    for (i = 1; i <= SIZE; i++) printf("%d\t%d\n", i, H[i]);
}
int main()
{
    int choice;
    do
    {
        cout << "Enter your choice\n";
        cout << " 1. Insert\n 2. Search\n 3. Display\n 0. Exit\n";
        cin >> choice;
        switch (choice)
        {
            case 1:
                Insert();

```

```
        display();
        break;
    case 2:
        Search();
        display();
        break;
    case 3:
        display();
        break;
    default:
        cout << "Enter correct choice\n";
        break;
    }
}
while (choice);
return 0;
}
```

Ans to the Question Number – 19

Ans to the Question Number – 20