**DSA ASSIGNMENT**

**Agnim Gupta**

**2028083**

**A-23, CSSE**

**Question 1**

#include<stdlib.h>

#include<stdio.h>

int main()

{

    printf("For the diagram - 01 :\n");

    for (int i = 1; i <= 3; i++)

    {

        for (int j = 1; j <= 3; j++)

        {

            if ((i == 1 && j == 2) || (i == 1 && j == 3) || (i == 2 && j == 3) || (i == 2 && j == 1))

            {

                printf("1 ");

            }

            else if ((i == 3 && j == 2) || (i == 3 && j == 1))

            {

                printf("1 ");

            }

            else

            {

                printf("0 ");

            }

        }

        printf("\n");

    }

    printf("For the digram - 02 :\n");

    for (int i = 1; i <= 3; i++)

    {

        for (int j = 1; j <= 3; j++)

        {

            if ((i == 1 && j == 2) || (i == 1 && j == 3) || (i == 2 && j == 3) || (i == 3 && j == 1))

            {

                printf("1 ");

            }

            else if ((i == 3 && j == 2))

            {

                printf("1 ");

            }

            else

            {

                printf("0 ");

            }

        }

        printf("\n");

    }

    printf("For the digram - 03 :\n");

    for (int i = 1; i <= 5; i++)

    {

        for (int j = 1; j <= 5; j++)

        {

            if ((i == 1 && j == 2) || (i == 2 && j == 1) || (i == 2 && j == 3) || (i == 3 && j == 2))

            {

                printf("1 ");

            }

            else if ((i == 3 && j == 4) || (i == 4 && j == 3) || (i == 3 && j == 5))

            {

                printf("1 ");

            }

            else if ((i == 5 && j == 3) || (i == 5 && j == 4) || (i == 4 && j == 5))

            {

                printf("1 ");

            }

            else

            {

                printf("0 ");

            }

        }

        printf("\n");

    }

    printf("For the digram - 04 :\n");

    for (int i = 1; i <= 5; i++)

    {

        for (int j = 1; j <= 5; j++)

        {

            if ((i == 1 && j == 2) || (i == 2 && j == 3) || (i == 3 && j == 4) || (i == 3 && j == 5))

            {

                printf("1 ");

            }

            else if (i == 4 && j == 5)

            {

                printf("1 ");

            }

            else

            {

                printf("0 ");

            }

        }

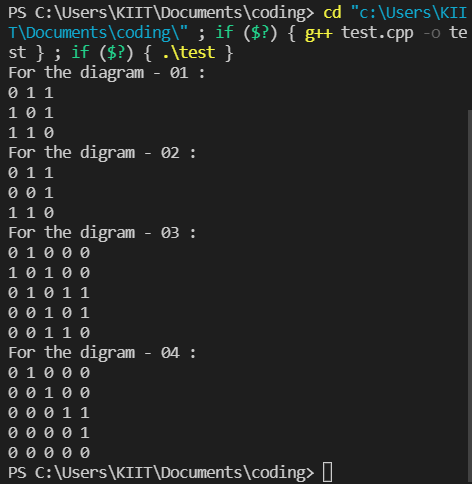
        printf("\n");

    }

    return 0;

}

**Output**



**Question 2**

#include <stdio.h>

#include <stdlib.h>

struct AdjListNode

{

    int dest;

    struct AdjListNode \* next;

};

struct AdjList

{

    struct AdjListNode \* head;

};

struct Graph

{

    int V;

    struct AdjList \* array;

};

struct AdjListNode \* newAdjListNode(int dest)

{

    struct AdjListNode \* newNode =(struct AdjListNode \* ) malloc(sizeof(struct AdjListNode));

    newNode -> dest = dest;

    newNode -> next = NULL;

    return newNode;

}

struct Graph \* createGraph(int V)

{

    struct Graph \* graph =(struct Graph \* ) malloc(sizeof(struct Graph));

    graph -> V = V;

    graph -> array =(struct AdjList \* ) malloc(V \* sizeof(struct AdjList));

    int i;

    for (i = 0; i < V; ++i)

    {

        graph -> array[i].head = NULL;

    }

    return graph;

}

void addEdge(struct Graph \* graph, int src, int dest)

{

    struct AdjListNode \* newNode = newAdjListNode(dest);

    newNode -> next = graph -> array[src].head;

    graph -> array[src].head = newNode;

    newNode = newAdjListNode(src);

    newNode -> next = graph -> array[dest].head;

    graph -> array[dest].head = newNode;

}

void printGraph(struct Graph \* graph)

{

    int v;

    for (v = 0; v < graph -> V; ++v)

    {

        struct AdjListNode \* pCrawl = graph -> array[v].head;

        printf("\n Adjacency list of vertex %d\n head ", v);

        while (pCrawl)

        {

            printf("-> %d", pCrawl -> dest); pCrawl = pCrawl -> next;

        }

        printf("\n");

    }

}

int main()

{

    int V = 5;

    struct Graph \* graph = createGraph(V);

    addEdge(graph,  0,  1);

    addEdge(graph,  0,  4);

    addEdge(graph,  1,  2);

    addEdge(graph,  1,  3);

    addEdge(graph,  1,  4);

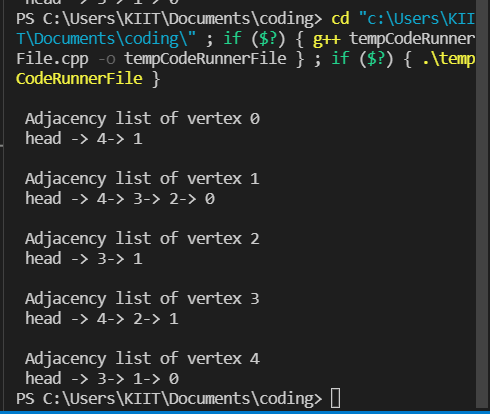
    addEdge(graph,  2,  3);

    addEdge(graph,  3,  4);

    printGraph(graph); return 0;

}

**Output**



**Question 3**

#include <stdio.h>

#include <stdlib.h>

struct nodes

{

int vertex; int edges;

int \*\* arr; // for creating 2-d arraay

};

void create\_graph(struct nodes \* temp, int x)

{

    temp -> vertex = x; temp -> edges = 0;

    temp -> arr = (int \*\* ) malloc(temp -> vertex \* sizeof(int \* ));

    for (int i = 0; i < temp -> vertex; i++)

    {

        temp -> arr[i] = (int \* ) malloc(temp -> vertex \* sizeof(int));

        for (int j = 0; j < temp -> vertex; j++)

        {

            temp -> arr[i][j] = 0;

        }

    }

}

void print\_graph(struct nodes \* t)

{

    for (int i = 0; i < t -> vertex; i++)

    {

        for (int j = 0; j < t -> vertex; j++)

        {

            printf("%d\t", t -> arr[i][j]);

        }

        printf("\n");

    }

}

void add\_edge(struct nodes \* t, int i, int j)

{

    if (i > 5 || j > 5)

    {

        printf("You have entered an wrong edge\n");

    }

    else

    {

        t -> arr[i - 1][j - 1] = 1;

    }

}

int main()

{

    struct nodes t; int v;

    printf("Enter the number of vertex present in the graph\n"); scanf("%d", & v);

    create\_graph( & t, v); add\_edge( & t, 1, 2);

    add\_edge( & t, 1, 3);

    add\_edge( & t, 1, 4);

    add\_edge( & t, 2, 3);

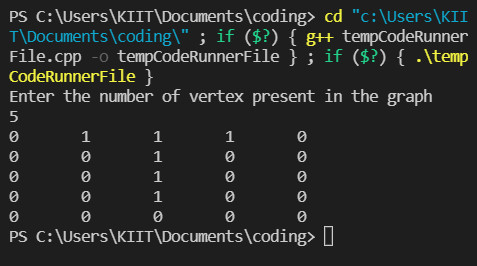
    add\_edge( & t, 3, 3);

    add\_edge( & t, 4, 3); print\_graph( & t);

    return 0;

}

**Output**



**Question 4**

#include <stdio.h>

#include <stdlib.h>

void insertion\_sort\_ascending(int A[], int n)

{

    for (int i = 1; i <= n; i++)

    {

        int j, k, x;

        j = i - 1;

        x = A[i];

        while (j > -1 && A[j] > x)

        {

            A[j + 1] = A[j]; j--;

        }

        A[j + 1] = x;

    }

}

void insertion\_sort\_decending(int A[], int n)

{

    for (int i = 1; i < n; i++)

    {

        int j, k, x;

        j = i - 1; x = A[i];

        while (j > -1 && A[j] < x)

        {

            A[j + 1] = A[j]; j--;

        }

        A[j + 1] = x;

    }

}

int main()

{

    int A[] = {1,2,3,4,5,6,7,8};

    int size\_1 = sizeof(A) / sizeof(A[0]);

    int B[] = {1,2,3,4,5,8,7,9,6};

    int size\_2 = sizeof(B) / sizeof(B[0]);

    int C[] = {2,1,4,3,6,5,7,8};

    int size\_3 = sizeof(C) / sizeof(C[0]);

    printf("CASE - 01 \n");

    insertion\_sort\_ascending(A, size\_1);

    for (int i = 0; i < size\_1; i++)

    {

        printf("%d\t", A[i]); printf("\n"); insertion\_sort\_decending(A, size\_1);

    }

    for (int i = 0; i < size\_1; i++)

    {

        printf("%d\t", A[i]);

    }

    printf("\n");

    printf("CASE - 02 \n");

    insertion\_sort\_ascending(B, size\_2);

    for (int i = 0; i < size\_2; i++)

    {

        printf("%d\t", B[i]); printf("\n");

    }

    insertion\_sort\_decending(B, size\_2);

    for (int i = 0; i < size\_2; i++)

    {

        printf("%d\t", B[i]); printf("\n"); printf("CASE - 03 \n");

    }

    insertion\_sort\_ascending(C, size\_3);

    for (int i = 0; i < size\_3; i++)

    {

        printf("%d\t", C[i]);

    }

    printf("\n"); insertion\_sort\_decending(C, size\_3);

    for (int i = 0; i < size\_3; i++)

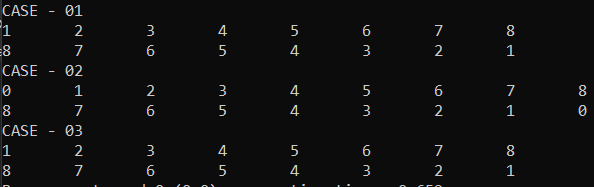
    {

        printf("%d\t", C[i]);

    }

    return 0;

}

**Output**

**Question 5**

#include<stdlib.h>

#include<stdio.h>

void swap(int \* a, int \* b, int \* s)

{

    int temp = \* a;

    \* a = \* b;

    \* b = temp; ( \* s) ++;

}

void selection\_sort(int A[], int n)

{

    int comp = 0, swaps = 0;

    for (int i = 0; i < n; i++)

    {

        int k;

        for (int j = k = i; j < n; j++)

        {

            if (A[j] < A[k])

            {

                k = j;

            }

            comp++;

        }

        swap( & A[i], & A[k], & swaps);

    }

    printf("The total number of possible comparisions are : %d\n", comp);

    printf("The total number of swaps are : %d\n", swaps);

}

int main()

{

    int A[] = {1,2,3,4,5,6,7,8};

    int size\_1 =sizeof(A) / sizeof(A[0]);

    int B[] = {1,2,3,4,5,8,7,9,6};

    int size\_2 =sizeof(B) / sizeof(A[0]);

    int C[] = {2,1,4,3,6,5,7,8};

    int size\_3 =sizeof(C) / sizeof(A[0]);

    printf("CASE    - 01: \n");

    selection\_sort(A, size\_1);

    for (int i = 0; i < size\_1; i++)

    {

        printf("%d\t", A[i]);

    }

    printf("\n");

    selection\_sort(A, size\_1);

    for (int i = size\_1 - 1; i >= 0; i--)

    {

        printf("%d\t", A[i]);

    }

    printf("\n");

    printf("CASE - 02: \n");

    selection\_sort(B, size\_2);

    for (int i = 0; i < size\_2; i++)

    {

        printf("%d\t", B[i]);

    }

    printf("\n");

    selection\_sort(B, size\_2);

    for (int i = size\_2 - 1; i >= 0; i--)

    {

        printf("%d\t", B[i]);

    }

    printf("\n");

    printf("CASE - 03: \n");

    selection\_sort(C, size\_3);

    for (int i = 0; i < size\_3; i++)

    {

        printf("%d\t", C[i]);

    }

    printf("\n");

    selection\_sort(C, size\_3);

    for (int i = size\_3 - 1; i >= 0; i--)

    {

        printf("%d\t", C[i]);

    }

    return 0;

}

**Output**

