

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

INTERNAL ASSIGNMENT

Course Code: OMC-307 July 23

Last Date of Submission: 31/12/24

Course Title: Design and Analysis of Algorithms Lab

Maximum Marks: 30

Session: July 2024

Deepankar Sharma

233512013

1. Write a C Program using recursive function to find the GCD of two numbers.

Program 01: GCD using Recursion

```
#include <stdio.h>
int gcd(int a, int b) {
    if(b == 0) return a;
    return gcd(b, a%b);
}
int main() {
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD of %d and %d is %d.", num1, num2, gcd(num1, num2));
    return 0;
}
```

Name: Deepankar Sharma
Course: MCA
Student ID: 233512013

Output

The screenshot shows a Visual Studio Code interface with the title bar "OMC307 Design and Analysis of Algorithms Laboratory". The left sidebar displays a file tree with several C programs (program_01.c through program_13.c) and a "Codes" folder containing "program_01.c" and "program_01.exe". The main editor window shows the code for "program_01.c", which includes a recursive function for finding the Greatest Common Divisor (GCD) and a main function that prompts the user for two numbers and prints their GCD. The terminal window on the right shows the command "deepa > OMC307 Design and Analysis of Algorithms Laboratory" followed by "cd .\Codes", "Codes\program_01.c", and "Codes\program_01.exe". It then prompts "Enter two numbers: 56 71" and outputs "GCD of 56 and 71 is 1".

```
#include <stdio.h>
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
int main() {
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD of %d and %d is %d\n", num1, num2, gcd(num1, num2));
    return 0;
}
```

This screenshot is nearly identical to the one above, showing the same Visual Studio Code interface and code for "program_01.c". The terminal output in the bottom right shows the same interaction with the user, prompting for "Enter two numbers: 56 71" and displaying the result "GCD of 56 and 71 is 1".

```
#include <stdio.h>
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
int main() {
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD of %d and %d is %d\n", num1, num2, gcd(num1, num2));
    return 0;
}
```

This screenshot shows a third execution of the same C program. The terminal output in the bottom right shows the user entering "16 32" and the program outputting "GCD of 16 and 32 is 16".

```
#include <stdio.h>
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
int main() {
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD of %d and %d is %d\n", num1, num2, gcd(num1, num2));
    return 0;
}
```

2. Write a C program using recursive function to implement towers of Hanoi problem.

```

Program 02 : Towers of Hanoi

#include <stdio.h>
void hanoi (int n, char from, char to, char aux) {
    if (n == 1) printf ("Move disk %c from %c to %c\n", from, to); return;
    hanoi (n-1, from, aux, to);
    printf ("Move disk %d from %c to %c\n", n, from, to);
    hanoi (n-1, aux, to, from);
}

int main() {
    int n;
    printf ("Enter the #disks: ");
    scanf ("%d", &n);
    hanoi (n, 'A', 'C', 'B');
    return 0;
}

```

Output

```

You, 22 hours ago | 1 author | 0 comments
Codes > C:\program_02\main
Enter the number of disks: 3
Move disk 1 From A to C
Move disk 2 From A to B
Move disk 1 From C to B
Move disk 3 From A to C
Move disk 1 From B to A
Move disk 2 From B to C
Move disk 1 From A to C

```

```

You, 22 hours ago | 1 author | 0 comments
Codes > C:\program_02\main
Enter the number of disks: 4
Move disk 1 From A to C
Move disk 2 From A to B
Move disk 1 From C to B
Move disk 3 From A to C
Move disk 1 From B to A
Move disk 2 From B to C
Move disk 1 From A to C
Move disk 4 From A to B
Move disk 1 From B to C
Move disk 2 From B to A
Move disk 1 From A to C
Move disk 3 From B to C
Move disk 1 From A to B
Move disk 2 From A to C
Move disk 1 From B to C

```

Output

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the title bar "OMCAT Design and Analysis of Algorithms Laboratory". The left sidebar displays a file tree with several C programs (e.g., program_01.c, program_02.c, etc.) and a "CODE" folder containing a "main.c" file. The main editor pane shows the "main.c" code for the Tower of Hanoi problem, which has been modified to solve for 5 disks. The terminal pane at the bottom right shows the output of the program, listing the steps to move 5 disks from A to C. The status bar at the bottom indicates "Launched" and provides a link to "deepl.com/main.c?main=main.c" for AI code review.

```
Code1 C program_01.c (main)
You, 22 hours ago | 1 author (v)
1 #include <stdio.h>
2
3 void hanoi(int n, char fr, char to, char aux) {
4     if (n == 1) {
5         printf("Move disk %d\n", n);
6         return;
7     }
8     hanoi(n - 1, fr, aux, to);
9     printf("Move disk %d\n", n);
10    hanoi(n - 1, aux, to, fr);
11 }
12
13 int main() {
14     int n;
15     printf("Enter the number of disks: ");
16     scanf("%d", &n);
17     hanoi(n, 'A', 'C', 'B');
18     return 0;
19 }
```

```
Move disk 2 From A to C
Move disk 1 From B to C
Codes \a.exe
Enter the number of disks: 5
Move disk 2 From A to C
Move disk 1 From C to B
Move disk 3 From A to C
Move disk 1 From B to A
Move disk 2 From B to C
Move disk 1 From A to C
Move disk 4 From A to B
Move disk 1 From C to B
Move disk 2 From C to A
Move disk 1 From B to C
Move disk 3 From A to B
Move disk 1 From A to C
Move disk 2 From A to B
Move disk 1 From C to B
Move disk 5 From A to C
Move disk 1 From B to A
Move disk 2 From B to C
Move disk 1 From A to C
Move disk 3 From B to A
Move disk 1 From C to B
Move disk 2 From A to C
Move disk 1 From B to A
Move disk 3 From B to C
Move disk 1 From A to C
Move disk 2 From A to B
Move disk 1 From C to B
Move disk 3 From A to C
Move disk 1 From B to A
Move disk 2 From B to C
Move disk 1 From A to C
```

3. Sort a given set of numbers using the merge sort method and determine the time required to sort them.

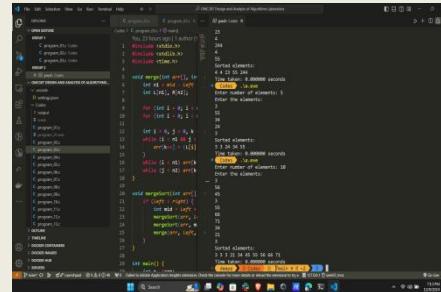
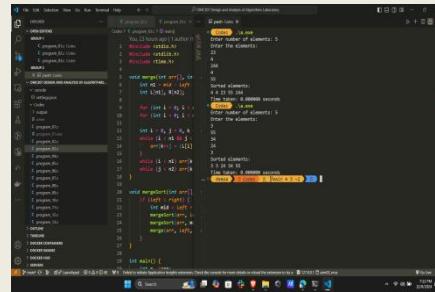
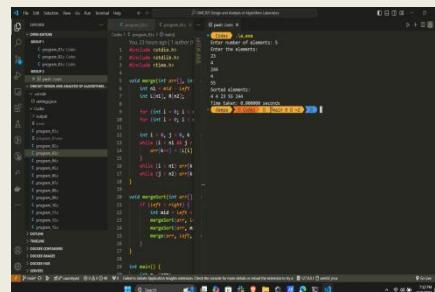
```
Program 03 : Merge Sort
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1, n2 = right - mid;
    int L[n1], R[n2];
    for (int i=0; i<n1; i++) L[i] = arr[left+i];
    for (int i=0; i<n2; i++) R[i] = arr[mid+1+i];
    int i=0, j=0, k=left;
    while (i<n1 && j<n2) {arr[k++] = (L[i]<= R[j])? L[i++]: R[j++];}
    while (j<n2) arr[k++] = R[j++];
    while (i<n1) arr[k++] = L[i++];
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + (right-left)/2);
        mergeSort(arr, left, mid);
        mergeSort(arr, mid+1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int n, *arr;
    printf("Enter #elements : ");
    scanf("%d", &n);
    arr = (int *) malloc(n * sizeof(int));
    for (int i=0; i<n; i++) scanf("%d", &arr[i]);
    mergeSort(arr, 0, n-1);
    printf("Sorted elements : ");
    for (int i=0; i<n; i++) printf("%d ", arr[i]);
    return 0;
}
```

Output



4. Sort a given set of numbers using the quick sort method and determine the time required to sort them.

Program 04 : MergeSort (QuickSort)

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

int partition (int arr[], int low, int high) {
    int pivot = arr[high], i = low - 1, temp;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            arr[i] = arr[j]; arr[j] = temp;
        }
    }
    temp = arr[++i]; arr[i] = arr[high];
    arr[high] = temp;
    return i;
}

void quickSort (int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main () {
    int n, *arr;
    printf ("Enter the #elements: ");
    scanf ("%d", &n);

    arr = (int *) malloc (n * sizeof(int));
    printf ("Enter the elements");
    for (int i = 0; i < n; i++) scanf ("%d", &arr[i]);

    quickSort(arr, 0, n - 1);

    printf ("sorted elements");
    for (int i = 0; i < n; i++) printf ("%d", arr[i]);
    return 0;
}
```

Name : Deepankar Sharma
 Course : MCA
 Student ID : 233512013

Output

```
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
13 void jobsequencingJob()
14 {
15     Enter the number of jobs: 5
16     for (int i = 0; i <= 4; i++)
17         for (int j = 0; j <= 4; j++)
18             if (select[j] == i)
19                 break;
20         }
21     }
22     Enter job id, deadline, and profit for Job 1: 1 2 3
23     Enter job id, deadline, and profit for Job 2: 2 3 4
24     Enter job id, deadline, and profit for Job 3: 3 4 5
25     Enter job id, deadline, and profit for Job 4: 4 5 6
26     Enter job id, deadline, and profit for Job 5: 5 6 8
27     Sorts elements: 1 2 3 4 5
28     Job sequence with maximum profit:
29     Job 1 (Profit: 3)
30     Job 2 (Profit: 4)
31     Job 3 (Profit: 5)
32     Job 4 (Profit: 6)
33     Job 5 (Profit: 8)
34     Job sequence: 5 4 3 2 1
35     profit("Job sequence: ")
36     for (int i = 0; i < 5; i++)
37         if (select[i] != -1)
38             cout<<"Job " << i+1 << endl;
39     }
40
41 int main()
42 {
43     int n;
44     printf("Enter the n: ");
45     scanf("%d", &n);
46
47     Job jobs[n];
48     for (int i = 0; i < n; i++)
49         printf("Enter j: ");
50     scanf("%d %d %d", &jobs[i].id, &jobs[i].deadline, &jobs[i].profit);
51
52     jobsequencing(jobs, n);
53     return 0;
54 }
```

```
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
13 void jobsequencingJob()
14 {
15     Enter Job Id, deadline, and profit for Job 1: 1 4 8
16     Enter Job Id, deadline, and profit for Job 2: 2 6 7
17     Enter Job Id, deadline, and profit for Job 3: 3 7 9
18     Enter Job Id, deadline, and profit for Job 4: 4 8 6
19     Enter Job Id, deadline, and profit for Job 5: 5 9 8
20     Sorts elements: 1 2 3 4 5
21     Job sequence with maximum profit:
22     Job 1 (Profit: 8)
23     Job 2 (Profit: 9)
24     Job 3 (Profit: 7)
25     Job 4 (Profit: 6)
26     Job 5 (Profit: 8)
27     Job sequence: 5 3 2 1 4
28     profit("Job sequence: ")
29     for (int i = 0; i < 5; i++)
30         if (select[i] != -1)
31             cout<<"Job " << i+1 << endl;
32     }
33
34     int main()
35     {
36         printf("Enter the n: ");
37         scanf("%d", &n);
38
39         Job jobs[n];
40         for (int i = 0; i < n; i++)
41             printf("Enter j: ");
42         scanf("%d %d %d", &jobs[i].id, &jobs[i].deadline, &jobs[i].profit);
43
44         jobsequencing(jobs, n);
45         return 0;
46     }
```

```
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
C:\program_03\src> C:\program_03\src> -> 37 push x          /| Codes | \.exe
13 void jobsequencingJob()
14 {
15     Enter the number of elements: 10
16     Enter the elements:
17     1
18     2
19     3
20     4
21     5
22     6
23     7
24     8
25     9
26     10
27     11
28     12
29     13
30     14
31     15
32     16
33     17
34     18
35     19
36     20
37     profit("Job sequence: ")
38     for (int i = 0; i < 20; i++)
39         if (select[i] != -1)
40             cout<<"Job " << i+1 << endl;
41
42     int main()
43     {
44         printf("Enter the n: ");
45         scanf("%d", &n);
46
47         Job jobs[n];
48         for (int i = 0; i < n; i++)
49             printf("Enter j: ");
50         scanf("%d %d %d", &jobs[i].id, &jobs[i].deadline, &jobs[i].profit);
51
52         jobsequencing(jobs, n);
53         return 0;
54     }
```

5. Given a set of n jobs, with (d_i) and (p_i) as deadline and profit for job i , write a program to find an optimal sequence of jobs, i.e., find the sequence of jobs that can be completed by their deadline and giving a maximum profit. Estimate the time complexity of the algorithm.

Output

```
C:\program_05> gcc -v <program_05.c
gcc.exe: error: <program_05.c: No such file or directory
gcc.exe: fatal error: no input files
compilation terminated.

C:\program_05> gcc -v <program_05.c
Enter the number of jobs: 5
Enter job id, deadline, and profit for job 1: 1 3 50
Enter job id, deadline, and profit for job 2: 4 5 60
Enter job id, deadline, and profit for job 3: 44 55 66
Enter job id, deadline, and profit for job 4: 44 3 55
Enter job id, deadline, and profit for job 5: 33 2 55
Maximum profit: 141
Job 1 (Profit: 55)
Job 3 (Profit: 55)
Job 4 (Profit: 66)
Job 44 (Profit: 66)
Job 4 (Profit: 45)

C:\program_05> .\a.exe
Enter the number of jobs: 5
Maximum profit: 141
Job 1 (Profit: 55)
Job 3 (Profit: 55)
Job 4 (Profit: 66)
Job 44 (Profit: 66)
Job 4 (Profit: 45)
```

Output

```
C:\program_05.c > jobSequencing(job[1], n)
13 void jobSequencing(job[1], n)
14 {
15     int i;
16     for (int i = 0; i < max; i++)
17     {
18         if (slot[i] != -1)
19         {
20             printf("Job %d is assigned to slot %d\n", i + 1, slot[i]);
21         }
22     }
23 }
24
25 int main()
26 {
27     int n;
28     printf("Enter the number of jobs: ");
29     scanf("%d", &n);
30
31     jobSequencing(job, n);
32     return 0;
33 }
```

```
Codes > gcc .\program_05
gcc.exe: error: .\program_05: No such file or directory
gcc.exe: fatal error: no input files
compilation terminated.

Codes > \.\program_05.c
Codes > \.\exe
Enter the number of jobs: 5
Enter job id, deadline, and profit for job 1: 1 3 55
Enter job id, deadline, and profit for job 2: 4 5 45
Enter job id, deadline, and profit for job 3: 4 4 55
Enter job id, deadline, and profit for job 4: 4 3 35
Enter job id, deadline, and profit for job 5: 3 2 55
Job sequence for maximum profit:
Job 1 (Profit: 55)
Job 33 (Profit: 55)
Job 3 (Profit: 55)
Job 44 (Profit: 66)
Job 4 (Profit: 45)

Codes > \.\exe
Enter the number of jobs: 4
Enter job id, deadline, and profit for job 1: 1 5 100
Enter job id, deadline, and profit for job 2: 2 2 23
Enter job id, deadline, and profit for job 3: 3 4 5
Enter job id, deadline, and profit for job 4: 4 5 6
4 5
Job sequence for maximum profit:
Job 1 (Profit: 100)
Job 2 (Profit: 23)
Job 3 (Profit: 5)
Job 3 (Profit: 55)
Job 1 (Profit: 100)
```

in push at 23:27:21

```
C:\program_05.c > jobSequencing(job[1], n)
13 void jobSequencing(job[1], n)
14 {
15     int i;
16     for (int i = 0; i < max; i++)
17     {
18         if (slot[i] != -1)
19         {
20             break;
21         }
22     }
23 }
24
25 int main()
26 {
27     int n;
28     printf("Enter the number of jobs: ");
29     scanf("%d", &n);
30
31     jobSequencing(job, n);
32     return 0;
33 }
```

```
Codes > \.\exe
Enter the number of jobs: 6
Enter job id, deadline, and profit for job 1: 1 2 3
Enter job id, deadline, and profit for job 2: 2 2 4
Enter job id, deadline, and profit for job 3: 3 4 5
Enter job id, deadline, and profit for job 4: 4 5 6
Enter job id, deadline, and profit for job 5: 5 6 83
Enter job id, deadline, and profit for job 6: 6 7 88
Job sequence for maximum profit:
Job 1 (Profit: 3)
Job 2 (Profit: 4)
Job 3 (Profit: 5)
Job 4 (Profit: 6)
Job 5 (Profit: 83)
Job 6 (Profit: 88)
```

in push at 23:31:52

6. Write C programs to find the minimum cost spanning tree using Kruskal's method.

Program 06 : Minimum Spanning Tree

```
#include <stdio.h>
#include <stdlib.h>
typedef struct { int u, v, weight; } Edge;
int find(int parent[], int i) {
    if (parent[i] == -1) return i;
    return find(parent, parent[i]);
}
void unionset(int parent[], int x, int y) { parent[x] = y; }
int compare(const void *a, const void *b) {
    return ((Edge *)a)->weight - ((Edge *)b)->weight;
}
void kruskal(Edge edges[], int n, int e) {
    int parent[n];
    for (int i=0; i<n; i++) parent[i] = -1;
    qsort(edges, e, sizeof(Edge), compare);
    printf("Edges in minimum spanning tree: ");
    int mincost = 0;
    for (int i=0; i<e; i++) {
        int x = find(parent, edges[i].u);
        int y = find(parent, edges[i].v);
        if (x != y) {
            printf("Edge(%d, %d) with (%d) weight \n", edges[i].u,
                   edges[i].v, edges[i].weight);
            mincost += edges[i].weight;
            unionset(parent, x, y);
        }
    }
    printf("Minimum cost: %d \n", mincost);
}
int main() {
    int n, e; printf("Enter # vertices and # edges: ");
    scanf("%d %d", &n, &e);
    Edge edges[e];
    for (int i=0; i<e; i++) {
        printf("Enter edge %d (u, v, weight): ", i+1);
        scanf("%d %d %d", &edges[i].u, &edges[i].v, &edges[i].weight);
    }
    kruskal(edges, n, e);
    return 0;
}
```

Name: Deepankar Sharma
 Course: MCA
 Student ID: 233512013

Output

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple files under "OPENED EDITS" including "program_06c.c".
- Code Editor:** Displays the C code for Kruskal's algorithm:

```
int kruskal(edge edges[], int n, int e) {
    int minCost = 0;
    qsort(edges, e, sizeof(edge), compare);
    printf("Edges in Minimum Spanning Tree:\n");
    for (int i = 0; i < e; i++) {
        int x = findParent(edges[i].u);
        int y = findParent(edges[i].v);
        if (x != y) {
            printf("%d (%d, %d) with weight %d\n", i, edges[i].u, edges[i].v, edges[i].weight);
            minCost += edges[i].weight;
            unionSet(parent, rank, x, y);
        }
    }
    return minCost;
}
```
- Terminal:** Shows the command to run the program: `cd "C:\Deeptapar\UCA\Semester 03\Assignments\GM307 Design and Analysis of Algorithms Laboratory\Codes"; if ($?) { gcc program_06c.c -o program_06c; } if ($?) { ./program_06c }`.
- Status Bar:** Shows the status message: "Teller file keybinding extension permission. Check the console for more details or revised the extension to tv-a" and the date "12/13/2024".

C program_06.c

```
for(j=0; j < n; j++) {  
    result[j] = -1;  
}  
int main() {  
    int n, e;  
    printf("Enter number of vertices and edges: ");  
    scanf("%d %d", &n, &e);  
    if(e > n - 1) {  
        printf("Error: Number of edges must be less than or equal to n - 1");  
        return 1;  
    }  
    edges = (Edge*)malloc(e * sizeof(Edge));  
    for(i = 0; i < e; i++) {  
        edges[i].v1 = i + 1;  
        edges[i].v2 = rand() % n + 1;  
        edges[i].weight = rand() % 10 + 1;  
    }  
    sort(edges, e);  
    minCost = 0;  
    for(i = 0; i < e; i++) {  
        if(result[edges[i].v1] == -1 && result[edges[i].v2] == -1) {  
            result[edges[i].v1] = edges[i].v2;  
            result[edges[i].v2] = edges[i].v1;  
            minCost += edges[i].weight;  
        }  
    }  
    printf("Edges in Minimum Spanning Tree:  
");  
    for(i = 0; i < e; i++) {  
        printf("%d (%d, %d), weight %d  
", i + 1, result[edges[i].v1], result[edges[i].v2], edges[i].weight);  
    }  
    printf("Minimum Cost: %d", minCost);  
}
```

C program_06_5

```
int main() {  
    int n, e;  
    printf("Enter number of vertices and edges: ");  
    scanf("%d %d", &n, &e);  
    if(e > n - 1) {  
        printf("Error: Number of edges must be less than or equal to n - 1");  
        return 1;  
    }  
    edges = (Edge*)malloc(e * sizeof(Edge));  
    for(i = 0; i < e; i++) {  
        edges[i].v1 = i + 1;  
        edges[i].v2 = rand() % n + 1;  
        edges[i].weight = rand() % 10 + 1;  
    }  
    sort(edges, e);  
    minCost = 0;  
    for(i = 0; i < e; i++) {  
        if(result[edges[i].v1] == -1 && result[edges[i].v2] == -1) {  
            result[edges[i].v1] = edges[i].v2;  
            result[edges[i].v2] = edges[i].v1;  
            minCost += edges[i].weight;  
        }  
    }  
    printf("Edges in Minimum Spanning Tree:  
");  
    for(i = 0; i < e; i++) {  
        printf("%d (%d, %d), weight %d  
", i + 1, result[edges[i].v1], result[edges[i].v2], edges[i].weight);  
    }  
    printf("Minimum Cost: %d", minCost);  
}
```

7. Write a C program to implement Dijkstra's Algorithm to find the shortest paths from a given source vertex to all other vertices.

Program 08: Dijkstra's Algorithm

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#define V 100

int minDistance(int dist[], bool optSet[], int vertices) {
    int min = INT_MAX, minIndex;
    for (int v = 0; v < vertices; v++) if (!optSet[v] && dist[v] <= min)
        min = dist[v], minIndex = v;
    return minIndex;
}

void dijkstra(int graph[V][V], int src, int vertices) {
    int dist[vertices]; bool optSet[vertices];
    for (int i = 0; i < vertices; i++) dist[i] = INT_MAX, optSet[i] = false;
    dist[src] = 0;
    for (int count = 0; count < vertices - 1; count++) {
        int u = minDistance(dist, optSet, vertices);
        optSet[u] = true;
        for (int v = 0; v < vertices; v++)
            if (!optSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) dist[v] = dist[u] + graph[u][v];
    }
    printf("Vertex \t Distance from source %d\n", src);
    for (int i = 0; i < vertices; i++) printf("%d \t %d\n", i, dist[i]);
}

int main() {
    int vertices; int src;
    printf("Enter # vertices: "); scanf("%d", &vertices);
    int graph[V][V];
    printf("Enter adjacency matrix: \n");
    for (int i = 0; i < vertices; i++)
        for (int j = 0; j < vertices; j++)
            scanf("%d", &graph[i][j]);
    scanf("%d", &src);
    dijkstra(graph, src, vertices);
    return 0;
}
```

Name : Deepankar Sharma
 Course : MCA
 Student ID : 233512013

Output

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple C files (program_1.c, program_2.c, etc.) and a folder named "CODES".
- Code Editor:** Displays the `program_08.c` file containing the Dijkstra's algorithm implementation.
- Terminal:** Shows the command line output of the program execution.
- Output:** Shows the results of the algorithm execution, including the adjacency matrix and shortest distances from source vertex 0.

```
14 void dijkstra(int graph[V][V], int src, int vertices) {
15     int dist[vertices];
16     bool sptSet[vertices];
17     for (int i = 0; i < vertices; i++)
18         dist[i] = INT_MAX, sptSet[i] = false;
19
20     dist[src] = 0;
21
22     for (int count = 0; count < vertices - 1; count++) {
23         int u = findMinDistNode(dist, sptSet);
24
25         for (int v = 0; v < vertices; v++)
26             if (graph[u][v] > 0 && !sptSet[v])
27                 dist[v] = min(dist[v], dist[u] + graph[u][v]);
28
29     }
30
31     printSolution(dist, vertices);
32 }
```

Terminal Output:

```
cd "C:\Deepankar\VMCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes"
if ($?) { ./program_08 }

Enter number of vertices: 4
Enter adjacency matrix:
0 0 4 0
0 0 0 3
0 0 0 0
Enter source vertex: 0
Vertex Distance from Source 0
0 0
1 1
2 4
3 5

in push at 20:32:39
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple C files (program_1.c, program_2.c, etc.) and a folder named "CODES".
- Code Editor:** Displays the `program_08.c` file containing the Dijkstra's algorithm implementation.
- Terminal:** Shows the command line output of the program execution.
- Output:** Shows the results of the algorithm execution, including the adjacency matrix and shortest distances from source vertex 0.

```
14 void dijkstra(int graph[V][V], int src, int vertices) {
15     int dist[vertices];
16     bool sptSet[vertices];
17     for (int i = 0; i < vertices; i++)
18         dist[i] = INT_MAX, sptSet[i] = false;
19
20     dist[src] = 0;
21
22     for (int count = 0; count < vertices - 1; count++) {
23         int u = findMinDistNode(dist, sptSet);
24
25         for (int v = 0; v < vertices; v++)
26             if (graph[u][v] > 0 && !sptSet[v])
27                 dist[v] = min(dist[v], dist[u] + graph[u][v]);
28
29     }
30
31     printSolution(dist, vertices);
32 }
```

Terminal Output:

```
cd "C:\Deepankar\VMCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes"
if ($?) { ./program_08 }

Enter number of vertices: 6
Enter adjacency matrix:
0 2 0 1 0 0
0 0 3 2 0 0
0 0 0 2 3 0
0 0 0 0 1 0
0 0 0 0 0 1
0 0 0 0 0 0
Enter source vertex: 0
Vertex Distance from Source 0
0 0
1 2
2 5
3 1
4 2
5 3

in push at 20:32:39
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple C files (program_1.c, program_2.c, etc.) and a folder named "CODES".
- Code Editor:** Displays the `program_08.c` file containing the Dijkstra's algorithm implementation.
- Terminal:** Shows the command line output of the program execution.
- Output:** Shows the results of the algorithm execution, including the adjacency matrix and shortest distances from source vertex 0.

```
14 void dijkstra(int graph[V][V], int src, int vertices) {
15     int dist[vertices];
16     bool sptSet[vertices];
17     for (int i = 0; i < vertices; i++)
18         dist[i] = INT_MAX, sptSet[i] = false;
19
20     dist[src] = 0;
21
22     for (int count = 0; count < vertices - 1; count++) {
23         int u = findMinDistNode(dist, sptSet);
24
25         for (int v = 0; v < vertices; v++)
26             if (graph[u][v] > 0 && !sptSet[v])
27                 dist[v] = min(dist[v], dist[u] + graph[u][v]);
28
29     }
30
31     printSolution(dist, vertices);
32 }
```

Terminal Output:

```
cd "C:\Deepankar\VMCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes"
if ($?) { ./program_08 }

Enter number of vertices: 9
Enter adjacency matrix:
9 12 0 2 3
7 10 11 0 1
0 5 99999 10
99999 0 3 99999
99999 99999 0 1
99999 99999 99999 0
Enter source vertex: 0
Vertex Distance from Source 0
0 0
1 5
2 8
3 9

in push at 20:32:39
```

8. Write C program to find all pairs of shortest paths using Floyd's algorithm.

Program 09 : Floyd's algorithm

```
#include <stdio.h>
#define INF 99999
#define V 100

void floyd_marshall (int graph[V][V], int vertices) {
    int dist[V][V];
    for (int i=0; i<vertices; i++) for (int j=0; j<vertices; j++)
        dist[i][j] = graph[i][j];
    for (int k=0; k<vertices; k++) for (int i=0; i<vertices; i++)
        for (int j=0; j<vertices; j++) {
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j] = dist[i][k] + dist[k][j];

    printf("All pairs shortest paths :\n");
    for (int i=0; i<vertices; i++) {
        for (int j=0; j<vertices; j++) {
            if (dist[i][j] == INF)
                printf("INF ");
            else
                printf("%d ", dist[i][j]);
        }
        printf("\n");
    }
}

int main () {
    int vertices;
    printf("Enter # vertices "); scanf("%d", &vertices);
    int graph[V][V];
    printf("Enter adjacency matrix ");
    for (int i=0; i<vertices; i++)
        for (int j=0; j<vertices; j++)
            scanf("%d ", &graph[i][j]);
    floyd_marshall(graph, vertices);
    return 0;
}
```

Name : Deepankar Sharma
 Course : MCA
 Student ID : 233512013

Output

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists several C programs and their corresponding executables (e.g., program_01.c, program_01.exe). The file "program_09.c" is currently selected.
- Open Editors:** Three C files are shown in the main editor area: "program_11.c", "program_10.c", and "program_09.c".
- Editor Content:** The code for "program_09.c" is displayed:

```
You, 5 minutes ago | author (You)
1 #include <stdio.h>
2
3 #define INF 99999
4 #define V 100
5
6 void floydWarshall(int graph[V][V], int vertices) {
7     int dist[V][V];
8     for (int i = 0; i < vertices; i++)
9         for (int j = 0; j < vertices; j++)
10            dist[i][j] = graph[i][j];
11 }
```
- Terminal:** The terminal window shows the execution of the program:

```
cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes" ; if ($?) { gcc pro
9.c -o program_09 } ; if ($?) { ./program_09 }
Enter number of vertices: 3
Enter adjacency matrix (use 99999 for INF):
0 1 99999
99999 0 1
1 99999 0
1 99999 0
All-Pairs Shortest Paths:
0 1 2
2 0 1
1 2 0
```
- Bottom Status Bar:** It displays the message "Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try a", along with other status indicators like battery level, signal strength, and file save status.

Output

9. Write C program to implement 0/1 knapsack problem using dynamic programming.

Program 10: 0-1 knapsack problem

Name : Deepankar Sharma
Course : MCA

```

#include <stdio.h>
int max(int a, int b) { return (a > b)? a : b; }
void knapsack(int W, int wt[], int val[], int n) {
    int K[n+1][W+1];
    for (int i=0; i<n; i++) for (int w=0; w<=W; w++) {
        if (i==0 || w==0) K[i][w]=0;
        else if (wt[i-1] <= w) K[i][w]=max(K[i-1][w],
                                              val[i-1] + K[i-1][w-wt[i-1]]);
        else K[i][w]=K[i-1][w];
    }
    printf("maximum value in knapsack = %d\n", K[n][W]);
}

```

Output

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of files and folders, with `C program_10.c` selected.
- Open Editors:** Displays two editor tabs: `C program_11.c M` and `C program_10.c, 6, M`.
- Code Editor:** The main area contains the following C code for a knapsack algorithm:

```
void knapsack(int W, int wt[], int val[], int n) {
    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= W; w++) {
            if (i == 0 || w == 0)
                K[i][w] = 0;
            else
                K[i][w] = K[i - 1][w];
            if (wt[i] <= w)
                K[i][w] = max(K[i][w], val[i] + K[i - 1][w - wt[i]]);
        }
    }
    printf("Maximum value in Knapsack = %d\n", K[n][W]);
}

int main() {
    int n, W;
```

- Terminal:** Shows the execution of the program and its output.
- Bottom Status Bar:** Includes icons for file operations, a search bar, taskbar, and system status.

Output

Screenshot of the Visual Studio Code interface. The left sidebar shows the file structure with several C programs and executables. The main editor window displays the code for program_10.c, which implements a knapsack algorithm using dynamic programming. The terminal window at the bottom shows the execution of the program and its output.

```
void knapsack(int W, int wt[], int val[], int n) {
    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= W; w++) {
            if (i == 0 || w == 0)
                K[i][w] = 0;
            else
                K[i][w] = K[i - 1][w];
            if (wt[i] <= w)
                K[i][w] = max(K[i][w], val[i] + K[i - 1][w - wt[i]]);
        }
    }
    printf("Maximum value in Knapsack = %d\n", K[n][W]);
}

int main() {
    int n, W;
    ...
```

TERMINAL OUTPUT:

```
cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if ($?) { gcc program_10.c -o program_10 } ; if ($?) { ./program_10 }
Enter number of items and capacity of knapsack: 2 5
Enter weights and values of items:
10 50
20 100
Maximum value in Knapsack = 0
```

VS Code status bar: Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again. 127.0.0.1 sem02_mca Spaces: 4 UTF-8 CRLF () C Go Live Win32

Screenshot of the Visual Studio Code interface, identical to the first one but with a different terminal input. The user has entered a different number of items and capacity, resulting in a different output.

```
cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if ($?) { gcc program_10.c -o program_10 } ; if ($?) { ./program_10 }
Enter number of items and capacity of knapsack: 3
50
Enter weights and values of items:
10 60
20 100
30 120
Maximum value in Knapsack = 220
```

Terminal message: in pwsh at 20:16:49

VS Code status bar: Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again. 127.0.0.1 sem02_mca Spaces: 4 UTF-8 CRLF () C Go Live Win32

10. Write a C program to perform the following operations on a graph:

- Depth First Search
- Breadth First Search

Problem 11 : Graph Traversal: BFS & DFS

```
void dfa (int graph[10][10], int visited[], int vertex, int n) {
    visited[vertex] = true; printf ("%d", vertex);
    for (int i=0; i<n; i++)
        if (graph[vertex][i] && !visited[i])
            dfa (graph, visited, i, n);
}

void bfa (int graph[10][10], int start, int n) {
    int visited[10] = {false}; int queue[10], front=0, rear=0;
    visited[start] = true; queue[rear++] = start;
    while (front < rear) {
        int vertex = queue[front++];
        printf ("%d", vertex);
        for (int i=0; i<n; i++) {
            if (graph[vertex][i] && !visited[i]) {
                visited[i] = true;
                queue[rear++] = i;
            }
        }
    }
}
```

Output

```
File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS CODES .vscode output a.exe nqueens.exe C_program_01.c C_program_01.exe C_program_02.c C_program_03.c C_program_04.c C_program_05.c C_program_06.c C_program_06.exe C_program_07.c C_program_08.c C_program_09.c C_program_10.c C_program_10.exe C_program_11.c M C_program_11.exe C_program_12.c M C_program_12.exe C_program_13.c M OUTLINE TIMELINE DOCKER CONTAINERS DOCKER IMAGES DOCKER HUB SERVERS Codes > main ~5 -1
program_11.c > main()
18 void bfs(int graph[][10], int start, int n) {
25     while (front < rear) {
28         for (int i = 0; i < n; i++) {
30             if (graph[vertex][i] && !visited[i]) {
31                 visited[i] = true;
32                 queue[rear++] = i;
33             }
34         }
35     }
36 }
38 int main() {
39     int n, graph[10][10];
40     printf("Enter the number of vertices: ");
1.c -o program_11 ; if ($?) { ./program_11 }
Enter the number of vertices: 5
Enter adjacency matrix:
0 1 0 0 1
1 0 1 0 0
0 1 0 1 0
0 0 1 0 1
1 0 0 1 0
DFS Traversal:
0 1 2 3 4
BFS Traversal:
0 1 4 2 3
in pwsh at 20:10:48
Spaces: 4 UTF-8 CRLF { } Go Live Win32
main* Launchpad Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again. 127.0.0.1 sem02_mca 8:12 PM 12/13/2024
```

Output

```
File Edit Selection View Go Run Terminal Help ← → ⌂ Codes ⌂ Codes program_11.c M X
C program_11.c > main()
18 void bfs(int graph[][10], int start, int n) {
25     while (front < rear) {
28         for (int i = 0; i < n; i++) {
30             if (graph[vertex][i] && !visited[i]) {
31                 visited[i] = true;
32                 queue[rear++] = i;
33             }
34         }
35     }
36 }
37
38 int main() {
39     int n, graph[10][10];
40     printf("Enter the number of vertices: ");
41
42     // Read input from terminal
43     // ...
44
45     // BFS Traversal
46     // ...
47
48     // DFS Traversal
49     // ...
50 }
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SPELL CHECKER DEBUG CONSOLE

Code + ⌂ Codes cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if (\$?) { gcc program_11 }
1.c -o program_11 ; if (\$?) { ./program_11 }
Enter the number of vertices: 6
Enter adjacency matrix:
0 1 0 0 0 0
1 0 1 1 0 0
0 1 0 0 1 0
0 1 0 0 0 1
0 0 1 0 0 1
0 0 1 0 0 1
DFS Traversal:
0 1 2 4 5 3

BFS Traversal:
0 1 2 3 4 5

Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again. 127.0.0.1 sem02_mca Spaces: 4 UTF-8 CRLF {} C Go Live Win32

8:11 PM 12/13/2024

```
File Edit Selection View Go Run Terminal Help ← → ⌂ Codes ⌂ Codes program_11.c M X
C program_11.c > main()
18 void bfs(int graph[][10], int start, int n) {
25     while (front < rear) {
28         for (int i = 0; i < n; i++) {
30             if (graph[vertex][i] && !visited[i]) {
31                 visited[i] = true;
32                 queue[rear++] = i;
33             }
34         }
35     }
36 }
37
38 int main() {
39     int n, graph[10][10];
40     printf("Enter the number of vertices: ");
41
42     // Read input from terminal
43     // ...
44
45     // BFS Traversal
46     // ...
47
48     // DFS Traversal
49     // ...
50 }
```

PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SPELL CHECKER DEBUG CONSOLE

Code + ⌂ Codes cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if (\$?) { gcc program_11 }
1.c -o program_11 ; if (\$?) { ./program_11 }
Enter the number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 1 1
1 1 0 1
0 1 1 0
DFS Traversal:
0 1 2 3

BFS Traversal:
0 1 2 3

Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again. 127.0.0.1 sem02_mca Spaces: 4 UTF-8 CRLF {} C Go Live Win32

8:11 PM 12/13/2024

11. Write a C program to find a subset ($S = \{s_1, s_2, \dots, s_m\}$) of a set (U) such that the sum of the elements of the subset (S) is equal to a given integer (d).

Program 12: subset sum

Name: Deepankar Sharma
Course: MCA

```

void subsetSum(int arr[], int subset[], int n, int index,
              int currentSum, int targetSum) {
    if (currentSum == targetSum) {
        for (int i=0; i < index; i++) printf("%d ", subset[i]);
        return;
    }
    if (n == 0) return;

    subset[index] = arr[0];
    subsetSum(arr+1, subset, n-1, index+1, currentSum+arr[0],
              targetSum);

    subsetSum(arr+1, subset, n-1, index, currentSum, targetSum);
}

```

output

File Edit Selection View Go Run Terminal Help

EXPLORER ... program_09.c C program_10.c C program_11.c C program_12.c C program_13.c C program_05.c C program_06.c C program_07.c C program_09.c C program_10.c C program_11.c C program_07.c C program... 2, M GROUP 2 C program... 2, M CODES

program_11.c > bfs(int |||10), int, int)

```

You, 6 days ago | 1 author (You)
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4
5 void dfs(int graph[][10], bool visited[], int vertex, int n)
6     visited[vertex] = true;
7     printf("%d ", vertex);
8
9     for (int i = 0; i < n; i++) {
10         if (graph[vertex][i] && !visited[i]) {
11             visited[i] = true;
12             dfs(graph, visited, i, n);
13         }
14     }

```

PROBLEMS 2 OUTPUT TERMINAL PORTS GITLENS AZURE SPELL CHECKER DEBUG CONSOLE

Codes cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if (\$?) { gcc program_12.c } ; if (\$?) { ./program_12 }

Enter the number of elements: 5

Enter the elements:

1 2 3 4 5

Enter the target sum: 5

Subsets with target sum:

1 4

2 3

5

in pwsh at 18:50:30

Spaces: 4 UTF-8 CRLF {} C Go Live Win32

output

The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows several C programs and their executables, such as `program_01.c`, `program_02.c`, etc., and `program_01.exe`, `program_02.exe`.
- Open Editors:** Displays multiple tabs of C code:
 - `program_07.c`: A function `main()` that initializes a graph and calls `prims(graph, vertices);`
 - `program_11.c`: A function `main()` that reads vertices and edges from input.
 - `program_07.c`: Another tab of the same program.
 - `program_12.c`: A function `printSubset` that prints subsets of a given size, and a function `subsetSum` that finds subsets with a target sum.
- Terminal:** Shows command-line interactions:
 - Execution of `program_12` followed by user inputs for number of elements (5) and target sum (5).
 - Execution of `program_12` followed by user inputs for number of elements (4) and target sum (5).
 - Execution of `program_12` followed by user inputs for number of elements (6) and target sum (20).
- Status Bar:** Shows the current file is `main.c`, the status bar message "Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again.", and the terminal output "in pwsh at 18:52:26".

The screenshot shows a Visual Studio Code interface with several tabs open in the Explorer view:

- C program_11.c
- C program_07.c 3
- C program_12.c 2
- C program_12.c M

The terminal window below shows the execution of a C program (program_07.c) which prints subsets of a given set of numbers. The user has entered 4 as the number of elements and 20 as the target sum. The output lists all subsets of {1, 2, 3, 4} that sum up to 20.

```
1 4
2 3
5
● Codes cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if ($?) { gcc program_12.c -o program_12 } ; if ($?) { ./program_12 }
Enter the number of elements: 4
Enter the elements:
1 2 6 8
Enter the target sum: 20
Subsets with target sum:
● Codes cd "c:\Deepankar\MCA\Semester 03\Assignments\OMC307 Design and Analysis of Algorithms Laboratory\Codes\" ; if ($?) { gcc program_12.c -o program_12 } ; if ($?) { ./program_12 }
Enter the number of elements: 4
Enter the elements:
1 2 3 4
Enter the target sum: 6
Subsets with target sum:
1 2 3
2 4
```

12. Write a C program to implement n-queen's problem using backtracking.

Program 13 : N Queens using Backtracking

```
#define N 8

bool isSafe(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++) if (board[row][i]) return false;
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j]) return false;
    for (int i = row, j = col; i < N && j >= 0; i++, j--)
        if (board[i][j]) return false;

    return true;
}

bool solveNQ(int board[N][N], int col) {
    if (col >= N) return true;
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            if (solveNQ(board, col + 1)) return true;
            board[i][col] = 0;
        }
    }
    return false;
}
```

Output

The screenshot shows a code editor with the following details:

- Code Editor Area:** Displays the C code for the N-Queens problem. The code includes functions for checking if a position is safe (`isSafe`), solving the problem for a given column (`solveNQ`), and printing the solution (`printSolution`). It also contains the `main` function which initializes the board and prints the solution for N=8.
- Terminal / Command Line:** Shows the command being run: `gcc .\program_13.c -o nqueens` followed by `.\nqueens.exe`. The output is labeled "N-Queens(N=8)" and shows the 8x8 board configuration with queens placed at (0,0), (1,4), (2,1), (3,6), (4,3), (5,0), (6,7), and (7,2).
- Status Bar:** At the bottom, it says "Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again." and shows the date and time as "12/9/2024 11:22 PM".

Output

The screenshot shows a Visual Studio Code interface with two panes. The left pane displays the C code for the N-Queens problem with N=8. The right pane shows the output of the program, which lists all 92 solutions for an 8x8 board. The output is a series of 8-digit binary strings representing column positions for each row.

```
1 0 0 0 0 0 0 0  
0 0 0 0 0 1 0  
0 0 0 1 0 0 0  
0 0 0 0 0 0 1  
0 1 0 0 0 0 0 0  
0 0 0 1 0 0 0 0  
0 0 0 0 0 1 0 0  
0 0 1 0 0 0 0 0  
N-Queens(N=8)  
1 0 0 0 0  
0 0 0 1 0  
0 1 0 0 0  
0 0 0 0 1  
0 0 1 0 0  
N-Queens(N=5)  
1 0 0 0 0  
0 0 0 1 0  
0 1 0 0 0  
0 0 0 0 1  
0 0 1 0 0  
N-Queens(N=5)
```

Below the panes, the status bar indicates "Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again." The system tray shows the date and time as 12/9/2024 at 11:23 PM.

The screenshot shows a Visual Studio Code interface with two panes. The left pane displays the C code for the N-Queens problem with N=10. The right pane shows the output of the program, which lists all 72 solutions for a 10x10 board. The output is a series of 10-digit binary strings representing column positions for each row.

```
1 0 0 0 0  
0 0 0 1 0  
0 1 0 0 0  
0 0 0 0 1  
0 0 1 0 0  
N-Queens(N=10)  
1 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 1 0 0  
0 1 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 1 0 0  
0 0 0 0 0 1 0 0 0 0  
0 0 1 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 1 0  
0 0 0 0 0 1 0 0 0 0  
0 0 0 0 1 0 0 0 0 0  
N-Queens(N=10)
```

Below the panes, the status bar indicates "Failed to initiate Application Insights extension. Check the console for more details or reload the extension to try again." The system tray shows the date and time as 12/9/2024 at 11:23 PM.