

EXPERIMENT 1


Min-Max Comparision

Code :

```
#include<stdio.h>
#include<stdlib.h>
int max;
int min;
int B[] = {4,5,46,2,6,55,26};
void minmax_DivCon(int A[], int i, int j)
{
    int mid;
    if (i == j) {
        max = (A[i] > max) ? A[i] : max;
        min = (A[i] < min) ? A[i] : min;
        return;
    }
    if (i == j - 1) {
        if (A[i] < A[j]) {
            min = (A[i] < min) ? A[i] : min;
            max = (A[j] > max) ? A[j] : max;
        } else {
            min = (A[j] < min) ? A[j] : min;
            max = (A[i] > max) ? A[i] : max;
        }
        return;
    }
    mid = (i + j) / 2;
    minmax_DivCon(A, i, mid);
    minmax_DivCon(A, mid + 1, j);
}
void minmax_naive(int A[], int size){
    if (size == 1) {
        min = A[0];
        max = A[0];
        return;
    }
    min = A[0];
    max = A[0];
    for (int i = 1; i < size; i++) {
        if (A[i] < min)
```

```
min = A[i];
if (A[i] > max)
max = A[i];
}
}
int main(){
printf("Vinit Shah C3-2 C183 600004220097\n");
max = B[0];
min = B[6];
minmax_DivCon(B, 0, 6);
printf("From: [4, 5, 46, 2, 6, 55, 26]\n");
printf("Maximum is %d\nMinimum is %d",max,min);
return 0;
}
```

Output:



```
Vinit Shah C3-2 C183 600004220097
From: [4, 5, 46, 2, 6, 55, 26]
Maximum is 55
Minimum is 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Strassen's Multiplication Algorithm

Code:

```
#include <stdio.h>
void naive_multiplication(int a[2][2],int b[2][2])
{
    int c[2][2],i,j,k;
    for (int i = 0; i <2 ; i++)
    {
        for (int j = 0; j <2; j++)
        {
            c[i][j] = 0;
            for (int k = 0; k <2; k++)
            {
                c[i][j] += a[i][k]*b[k][j];
            }
        }
    }
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            printf("%d ",c[i][j]);
        }
        printf("\n");
    }
}

void strassens(int a[2][2],int b[2][2])
{
    int p1,p2,p3,p4,p5,p6,p7,c00,c01,c10,c11;
    p1=a[0][0]*(b[0][1]-b[1][1]);
    p2=b[1][1]*(a[0][0]+a[0][1]);
    p3=b[0][0]*(a[1][0]+a[1][1]);
    p4=a[1][1]*(b[1][0]-b[0][0]);
    p5=(a[0][0]+a[1][1])*(b[0][0]+b[1][1]);
    p6=(a[0][1]-a[1][1])*(b[1][0]+b[1][1]);
    p7=(a[0][0]-a[1][0])*(b[0][0]+b[0][1]);
    c00=p4+p5+p6-p2;
    c01=p1+p2;
    c10=p3+p4;
    c11=p1-p3+p5-p7;
```

```
printf("\nBy Strassesns: \n");
printf("%d %d \n%d %d",c00,c01,c10,c11);
}
void main(){
printf("Vinit Shah C3-2 C183 60004220097\n");
int i,j,k,a[2][2],b[2][2];
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("Enter an element for Matrix A: ");
scanf("%d",&a[i][j]);
}
}
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("Enter an element for Matrix B: ");
scanf("%d",&b[i][j]);
}
}
naive_multiplication(a,b);
strassesns(a,b);
}
```

Output:

```
Vinit Shah C3-2 C183 60004220097
Enter an element for Matrix A: 1
Enter an element for Matrix A: 2
Enter an element for Matrix A: 3
Enter an element for Matrix A: 4
Enter an element for Matrix B: 5
Enter an element for Matrix B: 6
Enter an element for Matrix B: 7
Enter an element for Matrix B: 8
19 22
43 50

By Strassesns:
19 22
43 50

...Program finished with exit code 0
Press ENTER to exit console.
```

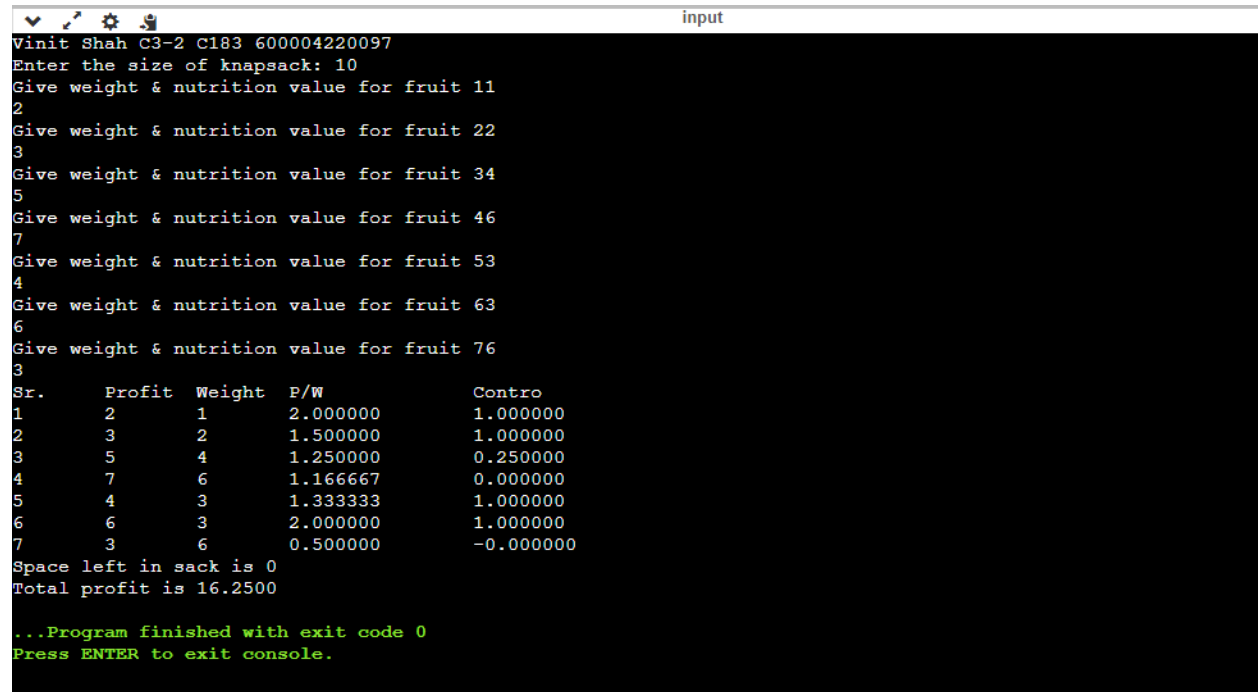
EXPERIMENT 2

Code :

```
#include<stdio.h>
#include<stdlib.h>
int main(){
printf("Vinit Shah C3-2 C183 600004220097\n");
int size = 7,i,j,temp,M;
float Max_prof = 0,ratio[size], accept[size];
int wt[size], profit[size], desc[size];
printf("Enter the size of knapsack: ");
scanf("%d",&M);
for(i=0;i<size;i++){
accept[size] = 0;
printf("Give weight & nutrition value for fruit %d",i+1);
scanf("%d%d",wt+i,profit+i);
ratio[i] = (float)profit[i]/wt[i];
desc[i] = i;
}
for (i = 0; i < size - 1; i++) {
for (j = 0; j < size - i - 1; j++) {if (ratio[desc[j]] < ratio[desc[j + 1]]) {
temp = desc[j];
desc[j] = desc[j+1];
desc[j+1] = temp;
}
}
}
for(i=0;i<size;i++){
if(M>wt[desc[i]]){
M -= wt[desc[i]];
accept[desc[i]] = 1;
}else{
accept[desc[i]] = (float)M/wt[desc[i]];
M = 0;
break;
}
}
printf("Sr.\tProfit\tWeight\tP/W\t\tContro\n");
for(i=0;i<size;i++){
printf("%d\t%d\t%d\t%f\t%f\n",i+1,profit[i],wt[i],ratio[i],accept[i]);
Max_prof += profit[i]*accept[i];
}
```

```
printf("Space left in sack is %d\n",M);  
printf("Total profit is %.4f",Max_prof);  
return 0;  
}
```

Output:



```
Vinit Shah C3-2 C183 600004220097  
Enter the size of knapsack: 10  
Give weight & nutrition value for fruit 11  
2  
Give weight & nutrition value for fruit 22  
3  
Give weight & nutrition value for fruit 34  
5  
Give weight & nutrition value for fruit 46  
7  
Give weight & nutrition value for fruit 53  
4  
Give weight & nutrition value for fruit 63  
6  
Give weight & nutrition value for fruit 76  
3  
3  
Sr.    Profit  Weight  P/W      Contro  
1      2       1       2.000000 1.000000  
2      3       2       1.500000 1.000000  
3      5       4       1.250000 0.250000  
4      7       6       1.166667 0.000000  
5      4       3       1.333333 1.000000  
6      6       3       2.000000 1.000000  
7      3       6       0.500000 -0.000000  
Space left in sack is 0  
Total profit is 16.2500  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

EXPERIMENT 3

Code :

```
#include<stdio.h>
#include<stdlib.h>
void swap(int *a, int *b){
int temp = *a;
*a = *b;
*b = temp;
}
int main(){
printf("Vinit Shah C3-2 C183 60004220097\n");
int size,i,j,u = 1,v = 0;
printf("Enter the number of jobs: ");
scanf("%d",&size);
int start[size],finish[size];
printf("Enter the start and finish times for Jobs:\n");
for(i=0;i<size;i++){
printf("Job %d",i+1);
scanf("%d%d",start+i,finish+i);
}
for (i = 0; i <size-1; i++) {
for (j = 0; j <size - i - 1; j++) {
if (finish[j] > finish[j + 1]) {
swap(finish+j, finish+ j + 1);
swap(start+j, start+ j + 1);
}
}
}
for(i=0;i<size;i++){
if(start[i] >= finish[v]){
u++;
v = i;
}
}
printf("Maximum number of non-conflicting tasks is %d",u);
return 0;
}
```

Output:

```
Vinit Shah C3-2 C183 60004220097
Enter the number of jobs: 5
Enter the start and finish times for Jobs:
Job 1
5
1
Job 22
5
Job 36
1
Job 42
4
Job 53
7
Maximum number of non-conflicting tasks is 4
```


EXPERIMENT 4

Code :

```
#include <stdio.h>
#include <limits.h>
#define vertices 6
int minDist(int dist[], int pre[]) {
    int minimum = INT_MAX, min, i;
    for (i = 0; i < vertices; i++)
        if (pre[i] == 0 && dist[i] <= minimum)
            minimum = dist[i], min = i;
    return min;
}
void printSolution(int dist[]) {
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < vertices; i++)
        printf("%d \t\t\t\t %d\n", i, dist[i]);
}
void dijkstra(int graph[vertices][vertices], int src)
{
    int dist[vertices];
    int pre[vertices];
    for (int i = 0; i < vertices; i++)
    {
        dist[i] = INT_MAX;
        pre[i] = 0;
    }
    dist[src] = 0;
    for (int count = 0; count < vertices - 1; count++)
    {
        int u = minDist(dist, pre);
        pre[u] = 1;
        for (int v = 0; v < vertices; v++)
            if (pre[v] == 0 && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] <
                dist[v])
                dist[v] = dist[u] + graph[u][v];
        printSolution(dist);
    }
}
int main() {
```

```
printf("Vinit Shah C3-2 C183 60004220097\n");
int graph[vertices][vertices] = {
{0, 3, 9, 5, 0, 0 },
{3, 0, 4, 3, 7, 0 },
{9, 4, 0, 2, 2, 2 },
{5, 3, 2, 0, 6, 8 },
{0, 7, 2, 6, 0, 5 },
{0, 0, 2, 8, 5, 0 }
};
dijkstra(graph, 0);
return 0;
}
```

Output:

input

```
Vinit Shah C3-2 C183 60004220097
Vertex      Distance from Source
0           0
1           3
2           9
3           5
4          2147483647
5          2147483647
Vertex      Distance from Source
0           0
1           3
2           7
3           5
4          10
5          2147483647
Vertex      Distance from Source
0           0
1           3
2           7
3           5
4          10
5           13
Vertex      Distance from Source
0           0
1           3
2           7
3           5
4           9
5           9
Vertex      Distance from Source
0           0
1           3
2           7
3           5
4           9
```

EXPERIMENT 5

Code:

```
#include <stdio.h>
int min(int a, int b) {
return (a < b) ? a : b;
}int minCoins(int coins[], int m, int V) {
int table[m + 1][V + 1];
for (int i = 0; i <= m; i++)
table[i][0] = 0;
for (int i = 1; i <= V; i++)
table[0][i] = i;
for (int i = 1; i <= m; i++) {
for (int j = 1; j <= V; j++) {
if (coins[i - 1] > j)
table[i][j] = table[i - 1][j];
else
table[i][j] = min(table[i - 1][j], 1 + table[i][j - coins[i - 1]]);
}
}
printf("Capacity/Coins\t\n");
printf("\t\t");
for (int i = 0; i <= V; i++) {
printf("%d\t", i);
}
printf("\n");
for (int i = 1; i <= m; i++) {
printf("%d\t\t", coins[i - 1]);
for (int j = 0; j <= V; j++) {
printf("%d\t", table[i][j]);
}
printf("\n");
}
int i = m, j = V, p = 0;
int sol[m + 1];while (j > 0) {
if (table[i][j] == table[i - 1][j])
i = i - 1;
else {
j = j - coins[i - 1];
sol[p] = coins[i - 1];
}
```

```
p++;
}
}
printf("Coins used: ");
for (int k = 0; k < p; k++) {
printf("%d ", sol[k]);
}
printf("\n");
return table[m][V];
}
int main() {
printf("Vinit Shah C3-2 C183 60004220097\n");
int coins[100], m, V;
printf("Enter the number of coins: ");
scanf("%d", &m);
printf("Enter the coins: ");
for (int i = 0; i < m; i++) {
scanf("%d", &coins[i]);
}
printf("Enter the total amount: ");
scanf("%d", &V);
int minCount = minCoins(coins, m, V);
printf("Minimum number of coins required: %d\n", minCount);return 0;
}
```

Output:

```
input
Vinit Shah C3-2 C183 60004220097
Enter the number of coins: 5
Enter the coins: 1
2
3
4
5
Enter the total amount: 12
Capacity/Coins
0      1      2      3      4      5      6      7      8      9      10     11     12
1      0      1      2      3      4      5      6      7      8      9      10     12
2      0      1      1      2      2      3      3      4      4      5      6      6
3      0      1      1      1      2      2      2      3      3      4      4      4
4      0      1      1      1      1      2      2      2      3      3      3      3
5      0      1      1      1      1      1      2      2      2      2      3      3
Coins used: 4 4 4
Minimum number of coins required: 3
```

EXPERIMENT 6

Code :

```
#include<stdio.h>
#include<stdlib.h>
int memo[10][10][2];
int MatrixChainOrder(int p[], int i, int j)
{
    if (i == j)
        return 0;
    if(memo[i][j][0] != 0){
        return memo[i][j][1];
    }
    int k;
    int min = 9999;
    int count;int k1;
    for (k = i; k < j; k++)
    {
        count = MatrixChainOrder(p, i, k) + MatrixChainOrder(p, k + 1, j) + p[i - 1] *
        p[k] *
        p[j];
        if (count < min){
            min = count;
            k1 = k;
        }
    }
    memo[i][j][0] = k1;
    memo[i][j][1] = min;
    return min;
}
int main(){
    printf("Vinit Shah C3-2 C183 60004220097\n");
    int n = 4,i,j;
    int cost[] = {5,4,2,3,4};
    for(i=0;i<10;i++){
        for(j=0;j<n;j++){
            memo[i][j][0] = 0;
            memo[i][j][1] = 0;
        }
    }
    printf("The original matrix is: \n");
    for(i=0;i<n;i++){
```

```
for(j=0;j<n;j++){
if(j>=i){
printf("%d ", MatrixChainOrder(cost, i+1, j+1));}else{
printf("0 ");
}
}
printf("\n");
}
printf("\n");printf("\nThe matrix chain order calculated is:\n");
for(i=0;i<n;i++){
for(j=0;j<n;j++){
if(j>=i){
printf("%d ", memo[i+1][j+1][0]);
}else{
printf("0 ");
}
}
printf("\n");
}
return 0;
}
```

Output:



```
Vinit Shah C3-2 C183 60004220097
The original matrix is:
0 40 70 104
0 0 24 56
0 0 0 24
0 0 0 0

The matrix chain order calculated is:
0 1 2 2
0 0 2 2
0 0 0 3
0 0 0 0

...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT 7

Code:

```
#include <stdio.h>
#include <stdlib.h>
void floydWarshall(int **graph, int n)
{
    int i, j, k;
    for (k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (graph[i][j] > graph[i][k] + graph[k][j])
                    graph[i][j] = graph[i][k] + graph[k][j];
            }
        }
    }
}
int main()
{
    printf("Vinit Shah C3-2 C183 60004220097\n");
    int n, i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    int **graph = (int **)malloc((long unsigned)n * sizeof(int *));
    for (i = 0; i < n; i++)
    {
        graph[i] = (int *)malloc((long unsigned)n * sizeof(int));
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (i == j)
                graph[i][j] = 0;
            else
                graph[i][j] = 100;
        }
    }
    printf("Enter the edges: \n");
    for (i = 0; i < n; i++)
```

```
{
for (j = 0; j < n; j++)
{printf("[%d][%d]: ", i, j);
scanf("%d", &graph[i][j]);
}
}
printf("The original graph is:\n");
for (i = 0; i < n; i++)
{
for (j = 0; j < n; j++)
{
printf("%d ", graph[i][j]);
}
printf("\n");
}
floydWarshall(graph, n);
printf("The shortest path matrix is:\n");
for (i = 0; i < n; i++)
{
for (j = 0; j < n; j++)
{
printf("%d ", graph[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:


```
Vinit Shah C3-2 C183 60004220097
Enter the number of vertices: 5
Enter the edges:
[0][0]: 1
[0][1]: 2
[0][2]: 3
[0][3]: 4
[0][4]: 5
[1][0]: 6
[1][1]: 7
[1][2]: 8
[1][3]: 9
[1][4]: 0
[2][0]: 1
[2][1]: 2
[2][2]: 3
[2][3]: 4
[2][4]: 5
[3][0]: 6
[3][1]: 7
[3][2]: 8
[3][3]: 9
[3][4]: 0
[4][0]: 1
[4][1]: 2
[4][2]: 3
[4][3]: 4
[4][4]: 5
The original graph is:
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
```

```
The original graph is:
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
The shortest path matrix is:
1 2 3 4 2
1 2 3 4 0
1 2 3 4 2
1 2 3 4 0
1 2 3 4 2

...Program finished with exit code 0
Press ENTER to exit console.
```

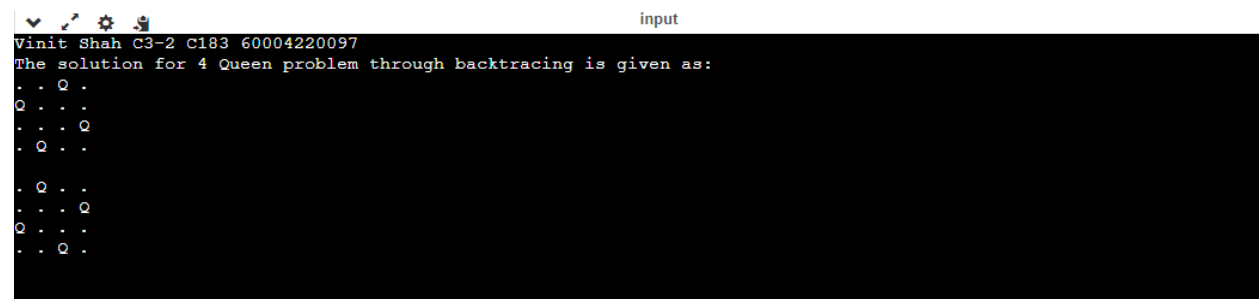
EXPERIMENT 7

Code:

```
#include <stdio.h>
#include <stdbool.h>
#define N 4
void printBoard(char board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf("%c ", board[i][j]);
        printf("\n");
    }
    printf("\n");
}
bool isSafe(char board[N][N], int row, int col) {
    int i, j;
    for (i = 0; i < col; i++)
        if (board[row][i] == 'Q')
            return false;
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j] == 'Q')
            return false;
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j] == 'Q')
            return false;
    return true;
}
bool solveNQueens(char board[N][N], int col) {
    if (col >= N) {
        printBoard(board);
        return true;
    }
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 'Q';
            solveNQueens(board, col + 1);
            board[i][col] = '.';
        }
    }
    return false;
}
void solveQueens() {
    char board[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            board[i][j] = '.';
        }
    }
}
```

```
solveNQueens(board, 0);  
}  
int main() {  
printf("Vinit Shah C3-2 C183 60004220097\n");  
printf("The solution for 4 Queen problem through backtracing is given as:\n");  
solveQueens();  
return 0;  
}
```

Output:



```
Vinit Shah C3-2 C183 60004220097  
The solution for 4 Queen problem through backtracing is given as:  
. . Q .  
Q . . .  
. . . Q  
. Q . .  
  
. Q . .  
. . . Q  
Q . . .  
. . Q .
```

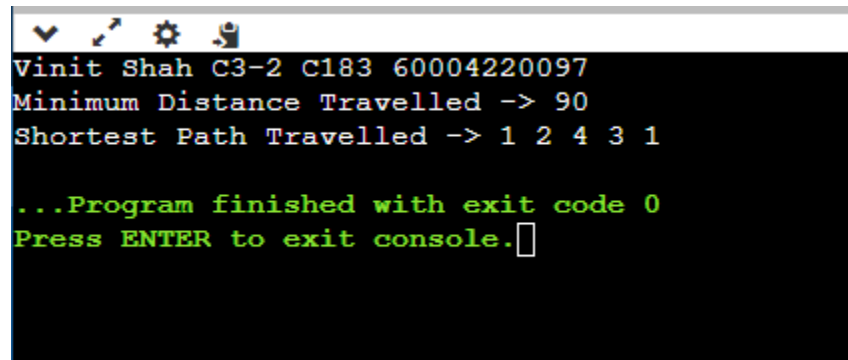
EXPERIMENT 9

Code:

```
#include <stdio.h>
#include <limits.h>
#include <math.h>
#define MAX 9999
int n = 4;int distan[20][20] = {
{ 0, 12, 18, 24 },
{ 12, 0, 36, 28 },
{ 18, 36, 0, 32 },
{ 24, 28, 32, 0 }};
int DP[16][4];
int TSP(int mark, int position) {
int completed_visit = pow(2, n) - 1;
if (mark == completed_visit) {
return distan[position][0];
}
if (DP[mark][position] != -1) {
return DP[mark][position];
}
int answer = MAX;
for (int city = 0; city < n; city++) {
if ((mark & (int)pow(2, city)) == 0) {
int newAnswer = distan[position][city] + TSP(mark | (int)pow(2, city), city);
answer = (answer < newAnswer) ? answer : newAnswer;
}
}
return DP[mark][position] = answer;
}
void printPath(int mark, int position) {
int completed_visit = pow(2, n) - 1;
if (mark == completed_visit) {
printf("%d ", position + 1);
return;
}int nextMark, city;
int minDist = MAX;
for (city = 0; city < n; city++) {
if ((mark & (int)pow(2, city)) == 0) {
int newMark = mark | (int)pow(2, city);
int newDist = distan[position][city] + TSP(newMark, city);
if (newDist == DP[mark][position]) {
nextMark = newMark;
break;
}
}
```

```
}  
}  
printf("%d ", position + 1);  
printPath(nextMark, city);  
}  
int main() {  
printf("Vinit Shah C3-2 C183 60004220097\n");  
for (int i = 0; i < pow(2, n); i++) {  
for (int j = 0; j < n; j++) {  
DP[i][j] = -1;  
}  
}  
printf("Minimum Distance Travelled -> %d\n", TSP(1, 0));  
printf("Shortest Path Travelled -> ");  
printPath(1, 0);  
printf("1");  
return 0;  
}
```

Output:



Vinit Shah C3-2 C183 60004220097
Minimum Distance Travelled -> 90
Shortest Path Travelled -> 1 2 4 3 1

...Program finished with exit code 0
Press ENTER to exit console.

EXPERIMENT 10

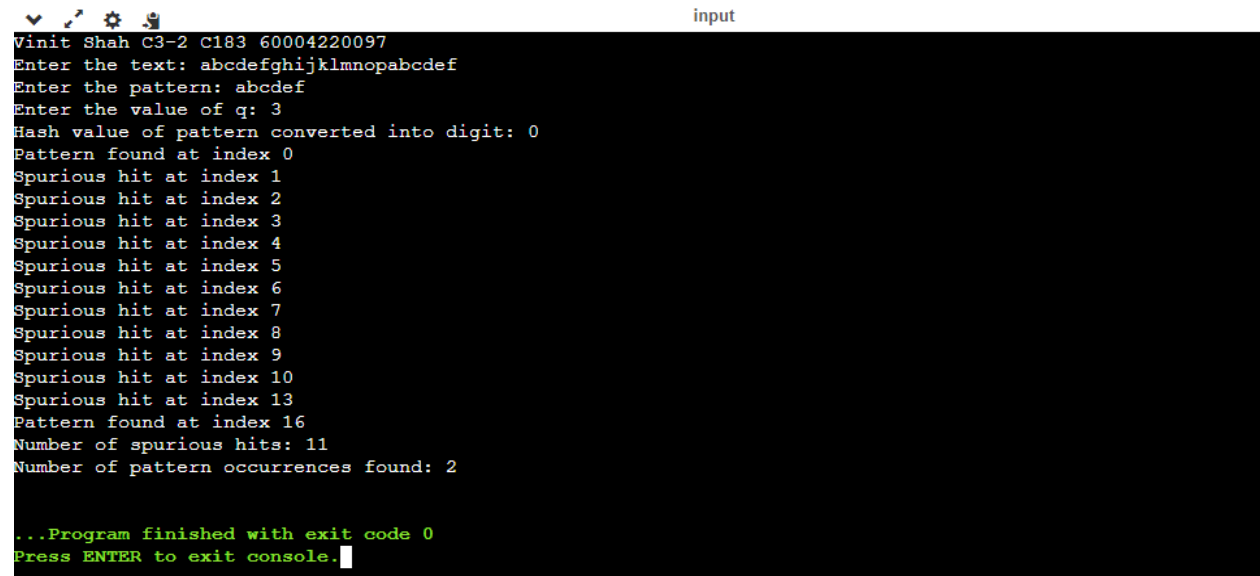
Code:

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

```
#include<string.h>
#include<math.h>
#define d 256
int hashToDigit(long long int hash, int q)
{
    return hash % q;
}
void search(char *pattern, char *text, int q)
{
    int M = strlen(pattern);
    int N = strlen(text);
    int i, j;
    int p = 0;
    int t = 0;
    int h = 1;
    int spuriousHits = 0;
    int patternCount = 0;
    for (i = 0; i < M - 1; i++)
        h = (h * d) % q;
    for (i = 0; i < M; i++)
    {
        p = (d * p + pattern[i]) % q; t = (d * t + text[i]) % q;
    }
    for (i = 0; i <= N - M; i++)
    {
        if (p == t)
        {
            for (j = 0; j < M; j++)
            {
                if (text[i + j] != pattern[j])
                    break;
            }
            if (j == M)
            {
                printf("Pattern found at index %d\n", i);
                patternCount++;
            }
            else
            {
                printf("Spurious hit at index %d\n", i);
                spuriousHits++;
            }
        }
    }
}
```

```
if (i < N - M)
{
t = (d * (t - text[i] * h) + text[i + M]) % q;
if (t < 0)
t = (t + q);
}
}printf("Number of spurious hits: %d\n", spuriousHits);
printf("Number of pattern occurrences found: %d\n", patternCount);
}
int main()
{
printf("Vinit Shah C3-2 C183 60004220097\n");
char text[100];
char pattern[100];
int q;
printf("Enter the text: ");
scanf("%s", text);
printf("Enter the pattern: ");
scanf("%s", pattern);
printf("Enter the value of q: ");
scanf("%d", &q);
int patternHashDigit = hashToDigit(123456789, q);
printf("Hash value of pattern converted into digit: %d\n", patternHashDigit);
search(pattern, text, q);
return 0;
}
```

Output:

A terminal window with a black background and white text. The window title is 'input'. The text shows the execution of a program that takes a text string, a pattern, and a value q as input. It calculates the hash of the pattern and finds occurrences of the pattern in the text, reporting spurious hits and the total number of occurrences.

```
Vinit Shah C3-2 C183 60004220097
Enter the text: abcdefghijklmnopabcdef
Enter the pattern: abcdef
Enter the value of q: 3
Hash value of pattern converted into digit: 0
Pattern found at index 0
Spurious hit at index 1
Spurious hit at index 2
Spurious hit at index 3
Spurious hit at index 4
Spurious hit at index 5
Spurious hit at index 6
Spurious hit at index 7
Spurious hit at index 8
Spurious hit at index 9
Spurious hit at index 10
Spurious hit at index 13
Pattern found at index 16
Number of spurious hits: 11
Number of pattern occurrences found: 2

...Program finished with exit code 0
Press ENTER to exit console.
```