DHRUV GARG   16BCE1190

Programme	:	**B.Tech (ECE|CSE)**	Semester	:	**FS 2017-18**
Course	:	**DATA STRUCTURES AND ALGORITHMS**	Code	:	**CSE2003**
Faculty	:	**Dr.Vetrivelan.P**	Slot	:	**G1**

Digital Assignment-#2 Questions (Implementation in C Programs)

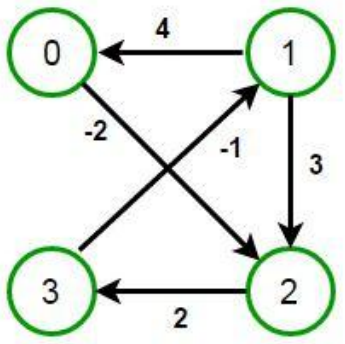
1. **Write a C Program to implement All-Pairs shortest path using Floyd's algorithm.**

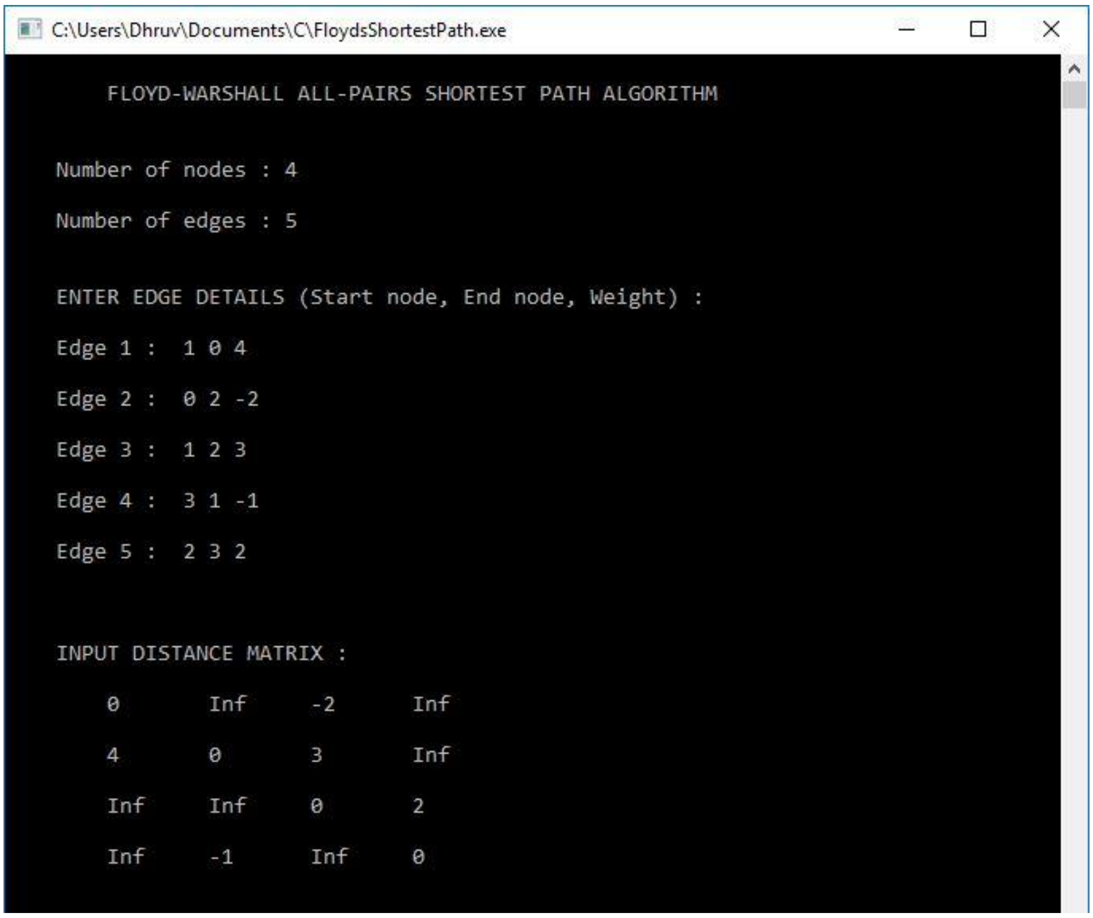
CODE :

```
#include<stdio.h>

int minimum(int a,int b)
{
   if(a < b)
      return a;
   else
      return b;
}

void floydWarshall(int dist[10][10],int nodes)
{
   int i, j, k;
   for (k=0; k < nodes; k++)
      for (i=0; i < nodes; i++)
         for (j=0; j < nodes; j++)
            if(i == j)
               dist[i][j]=0;
            else
               dist[i][j] = minimum(dist[i][j], dist[i][k]+dist[k][j]);
}
```

```
void main()
{
    int dist[10][10], nodes,edges;
    int start,end, weight, i = 0, j = 0;

    printf("\n\tFLOYD-WARSHALL ALL-PAIRS SHORTEST PATH ALGORITHM\n\n");
    printf("\n    Number of nodes : ");
    scanf("%d",&nodes);
    printf("\n    Number of edges : ");
    scanf("%d",&edges);
    printf("\n\n");

    for (i=0; i < nodes; i++)
    {
        for (j=0; j < nodes; j++)
            {
                if(i == j)
                    dist[i][j]=0;
                else
                    dist[i][j]=9999;
            }
    }

    printf("    ENTER EDGE DETAILS (Start node, End node, Weight) : \n\n");
    for (i=1; i <= edges; i++)
    {
        printf("    Edge %d :  ",i);
        scanf("%d%d%d",&start,&end,&weight);
        dist[start][end] = weight;
        printf("\n");
    }
    printf("\n\n    INPUT DISTANCE MATRIX : \n\n");
    for (i=0; i < nodes; i++)
    {
        printf("\t");
        for (j=0; j < nodes; j++)
        {
            if(dist[i][j] == 9999)
                printf("Inf \t");
            else
                printf("%d \t", dist[i][j]);
        }
        printf("\n\n");
    }
    printf("\n\n");
    floydWarshall(dist, nodes);
    printf("    SHORTEST-PATH DISTANCE MATRIX : \n\n");
    for (i=0; i < nodes; i++)
    {
        printf("\t");
        for (j=0; j < nodes; j++)
            printf("%d \t",dist[i][j]);
        printf("\n\n");
    }
    printf("\n\n");
    printf("    RESULT OF FLOYD-WARSHALL ALGORITHM : \n\n");
    for (i=0; i < nodes; i++)
        for (j=0; j < nodes; j++)
        {
            if(i != j)
                printf("\t( %d , %d ) = %d \n\n", i, j, dist[i][j]);
        }
}
```

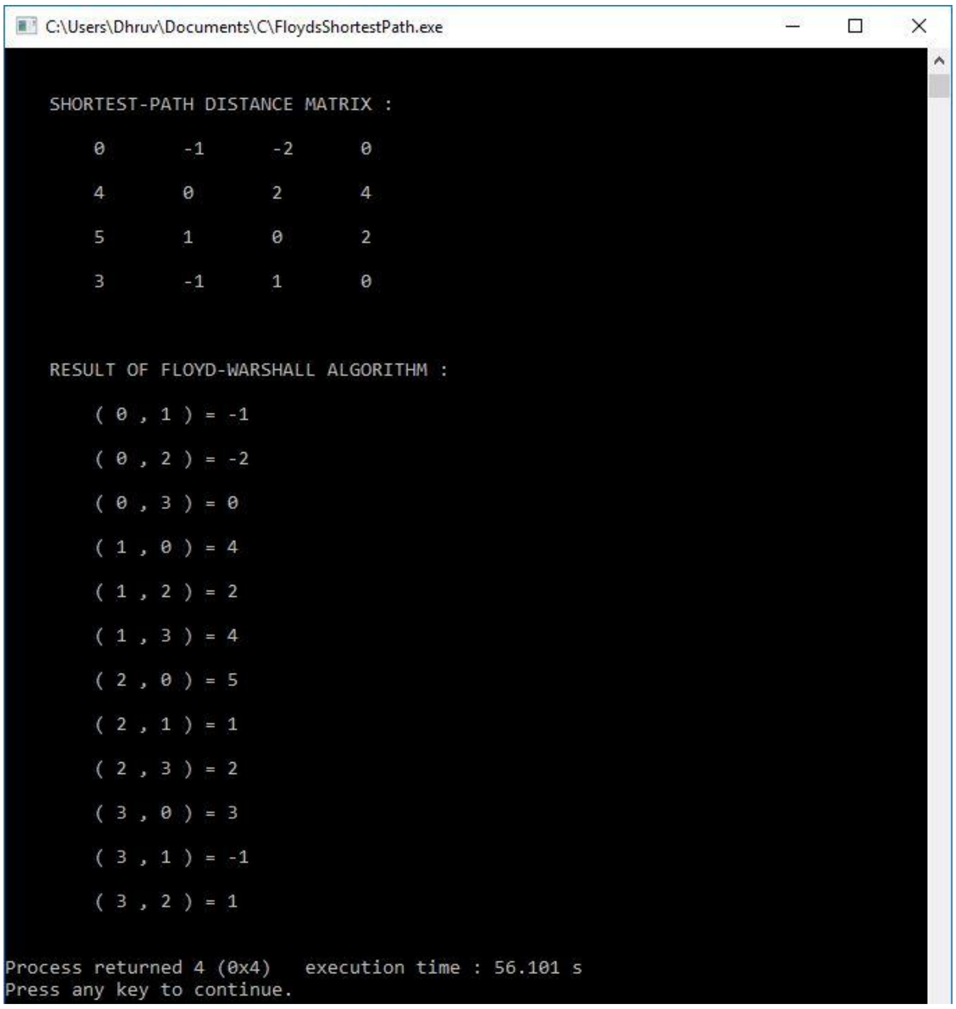
EXAMPLE 1





C:\Users\Dhruv\Documents\C\FloydsShortestPath.exe

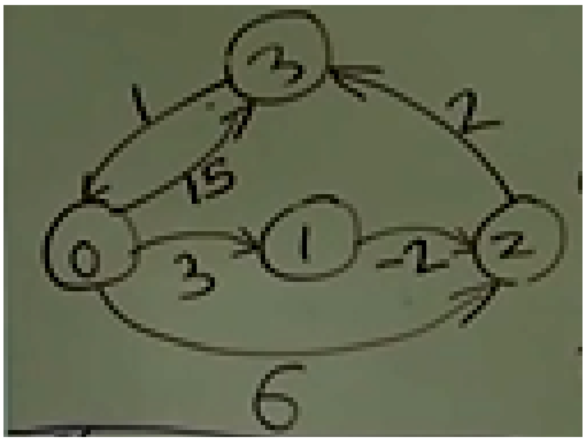
```
        FLOYD-WARSHALL ALL-PAIRS SHORTEST PATH ALGORITHM

    Number of nodes : 4

    Number of edges : 5

    ENTER EDGE DETAILS (Start node, End node, Weight) :

    Edge 1 :  1 0 4

    Edge 2 :  0 2 -2

    Edge 3 :  1 2 3

    Edge 4 :  3 1 -1

    Edge 5 :  2 3 2


    INPUT DISTANCE MATRIX :

        0       Inf     -2      Inf

        4       0       3       Inf

        Inf     Inf     0       2

        Inf     -1      Inf     0
```

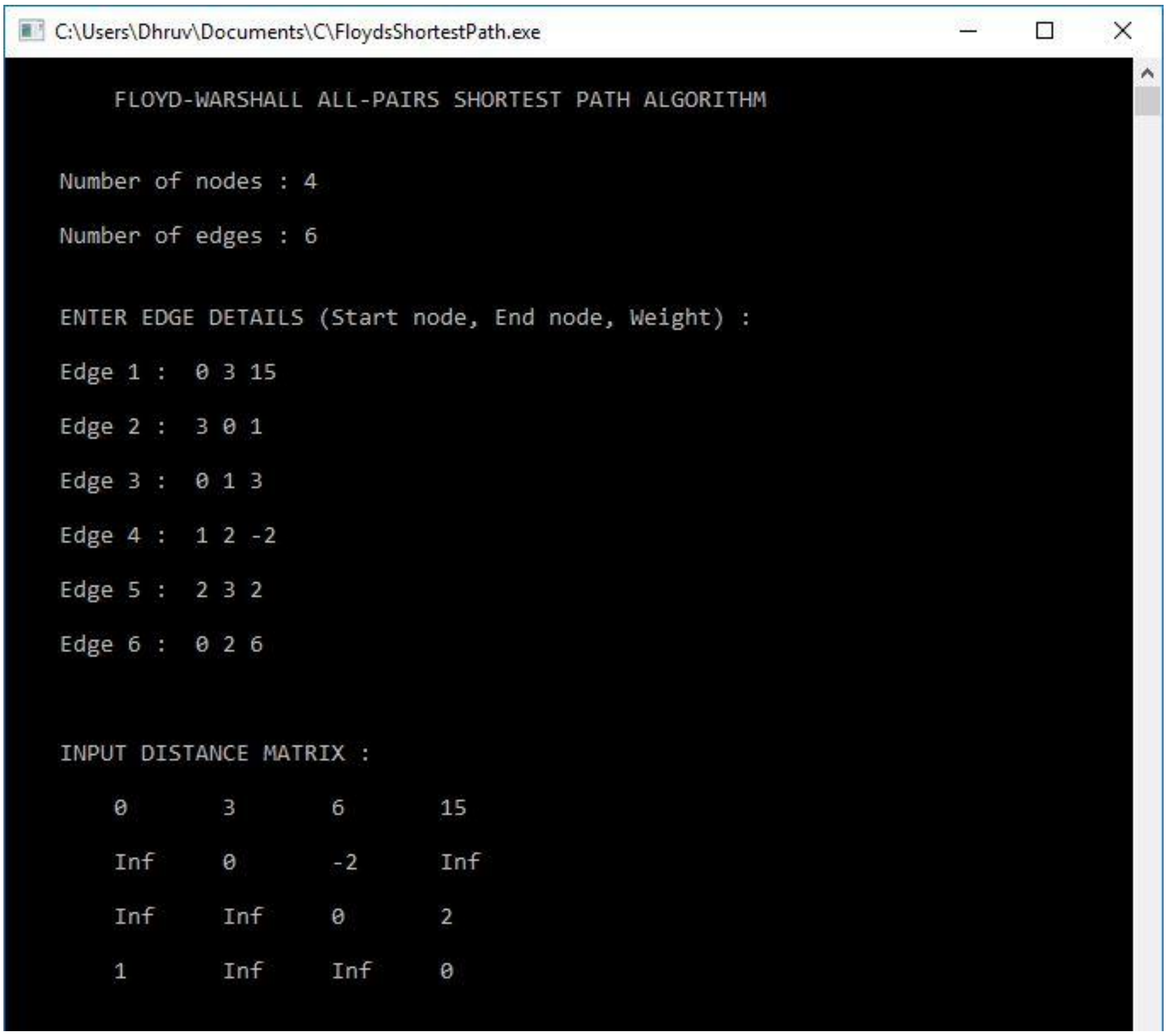


C:\Users\Dhruv\Documents\C\FloydsShortestPath.exe

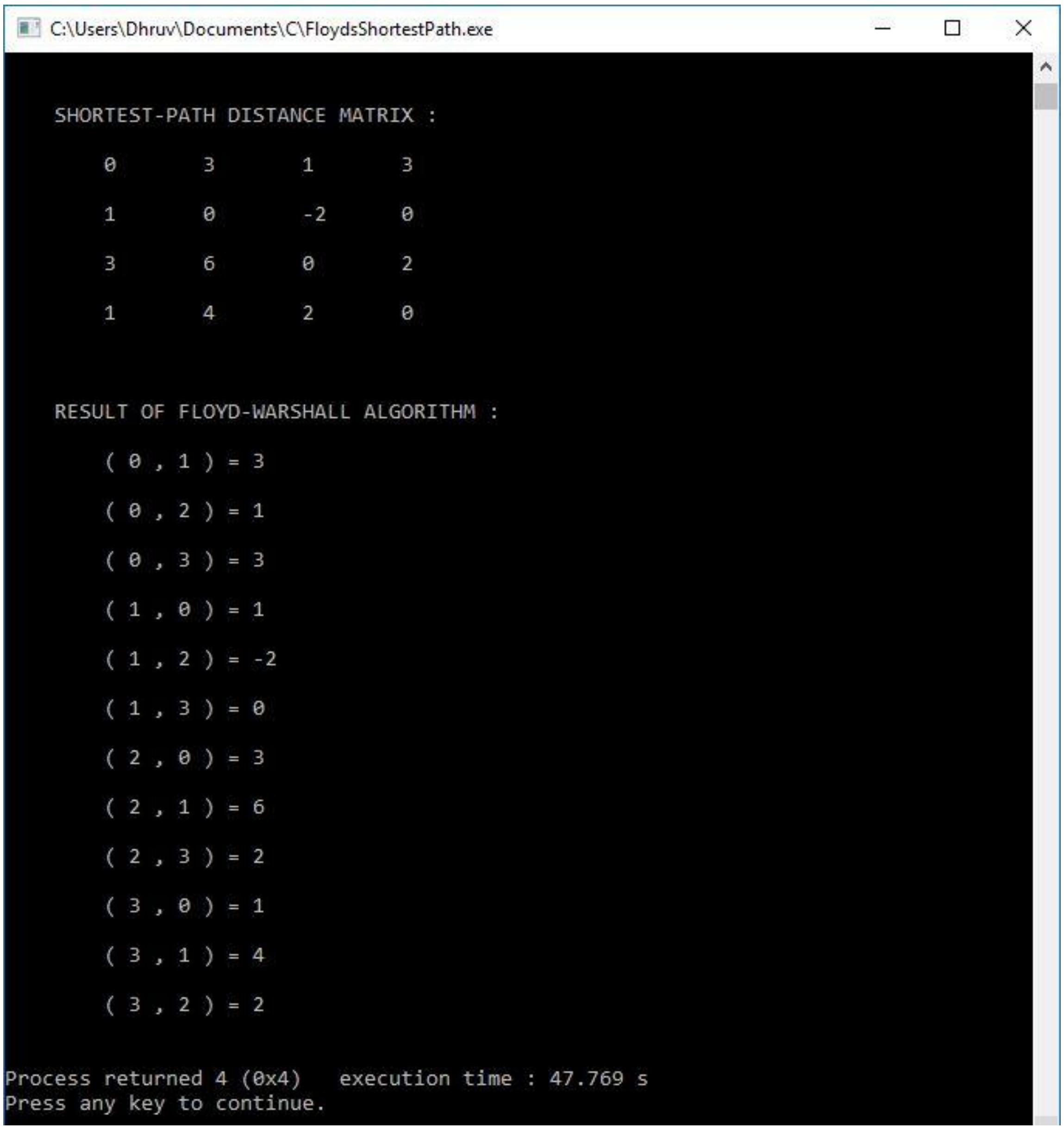
```
    SHORTEST-PATH DISTANCE MATRIX :

        0       -1      -2      0

        4       0       2       4

        5       1       0       2

        3       -1      1       0


    RESULT OF FLOYD-WARSHALL ALGORITHM :

        ( 0 , 1 ) = -1

        ( 0 , 2 ) = -2

        ( 0 , 3 ) = 0

        ( 1 , 0 ) = 4

        ( 1 , 2 ) = 2

        ( 1 , 3 ) = 4

        ( 2 , 0 ) = 5

        ( 2 , 1 ) = 1

        ( 2 , 3 ) = 2

        ( 3 , 0 ) = 3

        ( 3 , 1 ) = -1

        ( 3 , 2 ) = 1

Process returned 4 (0x4)   execution time : 56.101 s
Press any key to continue.
```

EXAMPLE 2





```
    FLOYD-WARSHALL ALL-PAIRS SHORTEST PATH ALGORITHM

Number of nodes : 4

Number of edges : 6

ENTER EDGE DETAILS (Start node, End node, Weight) :

Edge 1 :  0 3 15

Edge 2 :  3 0 1

Edge 3 :  0 1 3

Edge 4 :  1 2 -2

Edge 5 :  2 3 2

Edge 6 :  0 2 6


INPUT DISTANCE MATRIX :

    0       3       6       15

    Inf     0       -2      Inf

    Inf     Inf     0       2

    1       Inf     Inf     0
```



C:\Users\Dhruv\Documents\C\FloydsShortestPath.exe

```

    SHORTEST-PATH DISTANCE MATRIX :

        0       3       1       3

        1       0       -2      0

        3       6       0       2

        1       4       2       0


    RESULT OF FLOYD-WARSHALL ALGORITHM :

        ( 0 , 1 ) = 3

        ( 0 , 2 ) = 1

        ( 0 , 3 ) = 3

        ( 1 , 0 ) = 1

        ( 1 , 2 ) = -2

        ( 1 , 3 ) = 0

        ( 2 , 0 ) = 3

        ( 2 , 1 ) = 6

        ( 2 , 3 ) = 2

        ( 3 , 0 ) = 1

        ( 3 , 1 ) = 4

        ( 3 , 2 ) = 2


Process returned 4 (0x4)   execution time : 47.769 s
Press any key to continue.
```

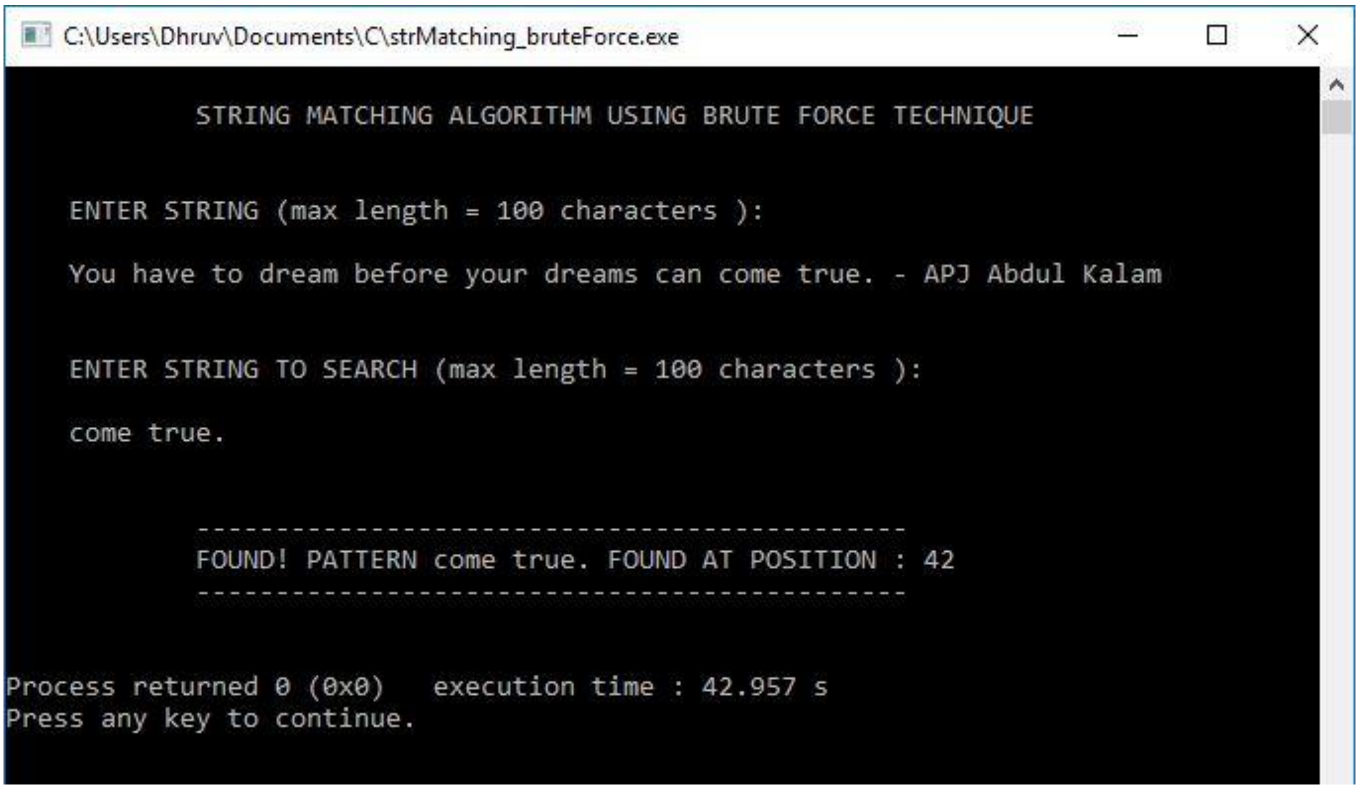
2. **String Matching Algorithm implementation in C using brute-force technique.**

CODE :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int pos = 0;
int bruteForceSearch(char *text, char *search)
{
   int i = 0, j = 0, flag = 0;
   int textLen = 0;
   int searchLen = 0;
   textLen = strlen(text);
   searchLen =strlen(search);
   for(i=0; i <= textLen-searchLen; i++)
   {
      j=0;
      while(j<searchLen && search[j]==text[j+i])
      {
         j++;
         if(j==searchLen)
         {
            flag=1;
            pos=i+1;
            return 1;
         }
         else
            flag=0;
      }
   }
   return 0;
}

void main()
{
   int i,j, result=0;
   char text[50], search[50];
```

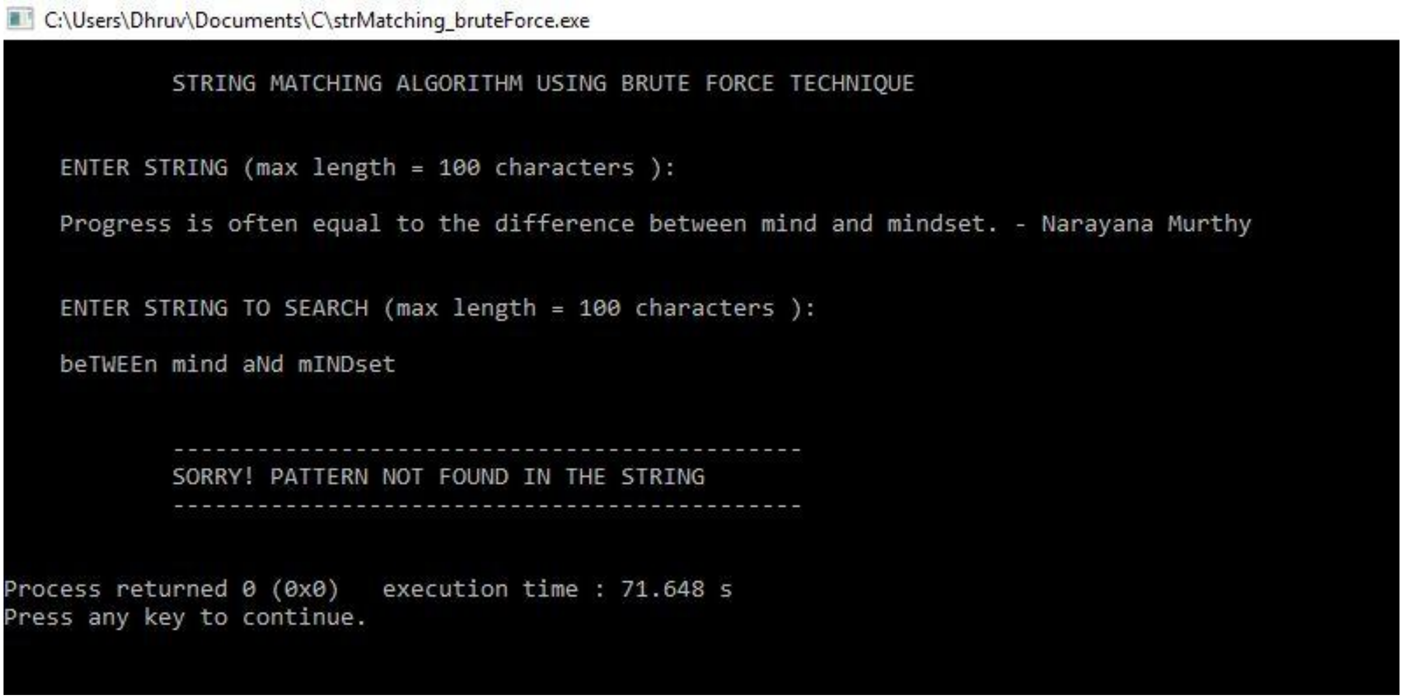
```
    printf("\n\t    STRING MATCHING ALGORITHM USING BRUTE FORCE TECHNIQUE \n\n\n");
    printf("    ENTER STRING (max length = 50 characters ): \n\n    ");
    gets(text);
    printf("\n\n");
    printf("    ENTER STRING TO SEARCH (max length = 50 characters ): \n\n    ");
    gets(search);
    result = bruteForceSearch(text, search);
    if(result==1)
    {
        printf("\n\n\t    --------------------------------------------\n");
        printf("\t    FOUND! PATTERN %s FOUND AT POSITION : %d", search, pos);
        printf("\n\t    --------------------------------------------\n\n");
    }
    else
    {
        printf("\n\n\t    --------------------------------------------\n");
        printf("\t    SORRY! PATTERN NOT FOUND IN THE STRING");
        printf("\n\t    --------------------------------------------\n\n");
    }
}
```

EXAMPLE 1



```
C:\Users\Dhruv\Documents\C\strMatching_bruteForce.exe

            STRING MATCHING ALGORITHM USING BRUTE FORCE TECHNIQUE


    ENTER STRING (max length = 100 characters ):

    You have to dream before your dreams can come true. - APJ Abdul Kalam


    ENTER STRING TO SEARCH (max length = 100 characters ):

    come true.



            --------------------------------------------
            FOUND! PATTERN come true. FOUND AT POSITION : 42
            --------------------------------------------


Process returned 0 (0x0)   execution time : 42.957 s
Press any key to continue.
```

EXAMPLE 2



C:\Users\Dhruv\Documents\C\strMatching_bruteForce.exe

```
            STRING MATCHING ALGORITHM USING BRUTE FORCE TECHNIQUE


    ENTER STRING (max length = 100 characters ):

    Progress is often equal to the difference between mind and mindset. - Narayana Murthy


    ENTER STRING TO SEARCH (max length = 100 characters ):

    beTWEEn mind aNd mINDset


            ---------------------------------------------
            SORRY! PATTERN NOT FOUND IN THE STRING
            ---------------------------------------------


Process returned 0 (0x0)   execution time : 71.648 s
Press any key to continue.
```