Assignment 10: Implementation of Shortest Path Finding algorithm

The cityADT contains the number of cities and the connectivity information between the cities (adjacency matrix). Write the following methods.

- void create(cityADT *C) – will represent the graph using adjacency matrix
- void disp(cityADT *C) – Display the graph
- void Dijkstra(cityADT *C)
  – Displays the intermediate and final tables
- char * displayPath(cityADT *C, source, destination)
  – Find the path of the intermediate cities between the source and destination cities along with the cost

CODE:

Dijkstras.h

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

int main() {
  struct graph *g;
  g = (struct graph *)malloc(sizeof(struct graph));
  int vertex, edges;
  int d;
  printf("enter the number of vertex:");
  scanf("%d", &vertex);
  printf("enter the number of edges:");
  scanf("%d", &edges);
  struct edgePair edge[100];
  int x = 1;

  while (x) {
    printf("Enter:\n1.create\n2.display\n3.printTable\n4.shotest "
           "path\n5.path\n6.exit");
    int choice, s;
    int from, to, weight;
    printf("choice the option\n");
    scanf("%d", &choice);
```

```
switch (choice)

{

case 1:

  printf("enter the edge pairs\n");
  for (int i = 0; i < edges; i++) {
    scanf("%d", &edge[i].from);
    scanf("%d", &edge[i].to);

    // scanf("%d",&edge[i].weight);
  }
  create(g, vertex, edges, edge);
  printf("\n");
  break;

case 2:

  printf("ADJENCY MATRICES;\n");
  display(g);
  printf("\n");
  break;

case 3:

  init(g);
  printf("TABLE:\n");
  printTable(g,vertex);
  printf("\n");
  break;
case 4:

  printf("enter the starting vertex:\n");
  scanf("%d", &s);
  dijkstras(g, s,vertex);
  printf("\n");
  break;
case 5:

  printf("Enter the destination: ");
  scanf("%d", &d);
  printf("PATH: \n");
```

```
      path(g, d);
      printf("\n");
      break;
    case 6:
      x = 0;
      printf("exit\n");
      break;
    }
  }
}
```

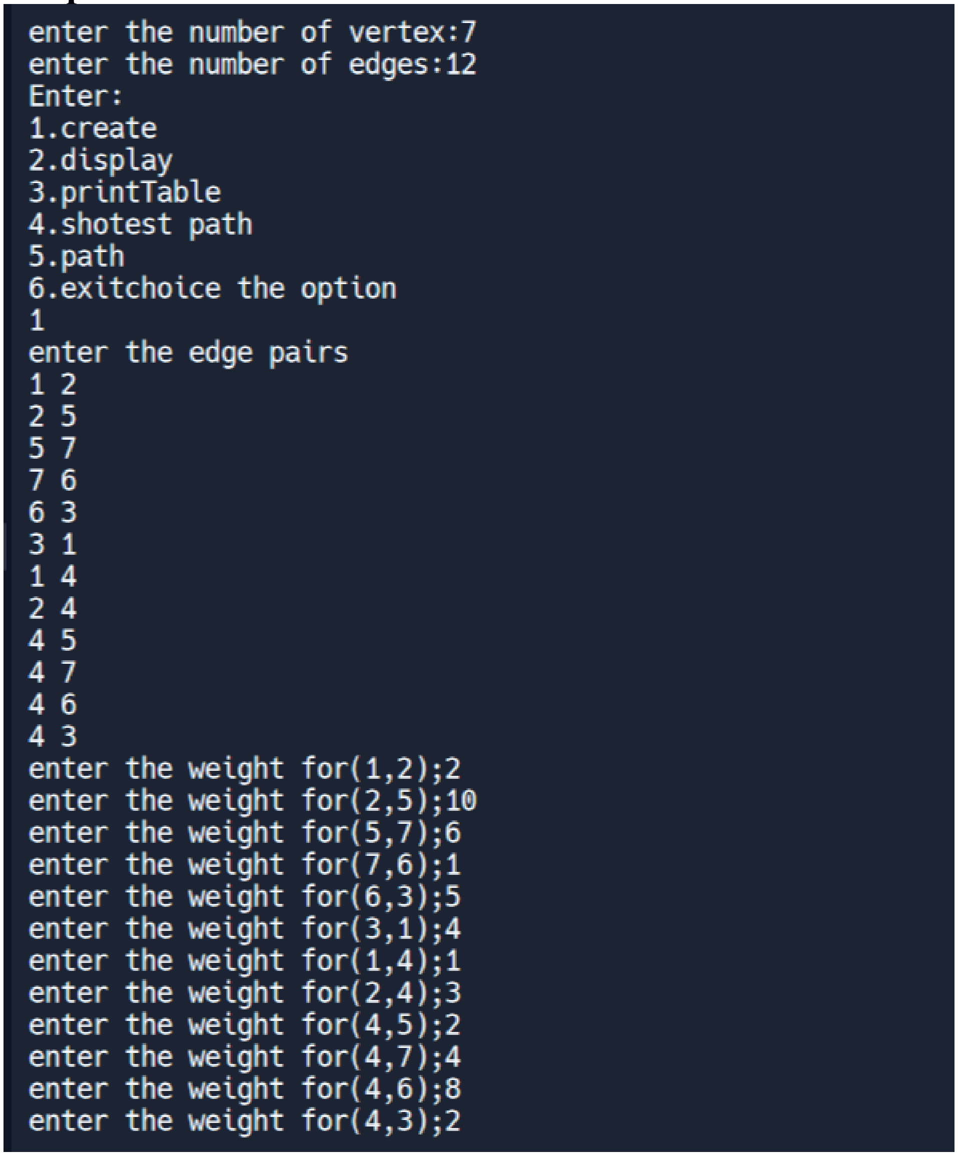
Main.c

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

int main() {
  struct graph *g;
  g = (struct graph *)malloc(sizeof(struct graph));
  int vertex, edges;
  int d;
  printf("enter the number of vertex:");
  scanf("%d", &vertex);
  printf("enter the number of edges:");
  scanf("%d", &edges);
  struct edgePair edge[100];
  int x = 1;

  while (x) {
    printf("Enter:\n1.create\n2.display\n3.printTable\n4.shotest "
           "path\n5.path\n6.exit");
    int choice, s;
    int from, to, weight;
    printf("choice the option\n");
    scanf("%d", &choice);

    switch (choice)

    {

    case 1:

      printf("enter the edge pairs\n");
```

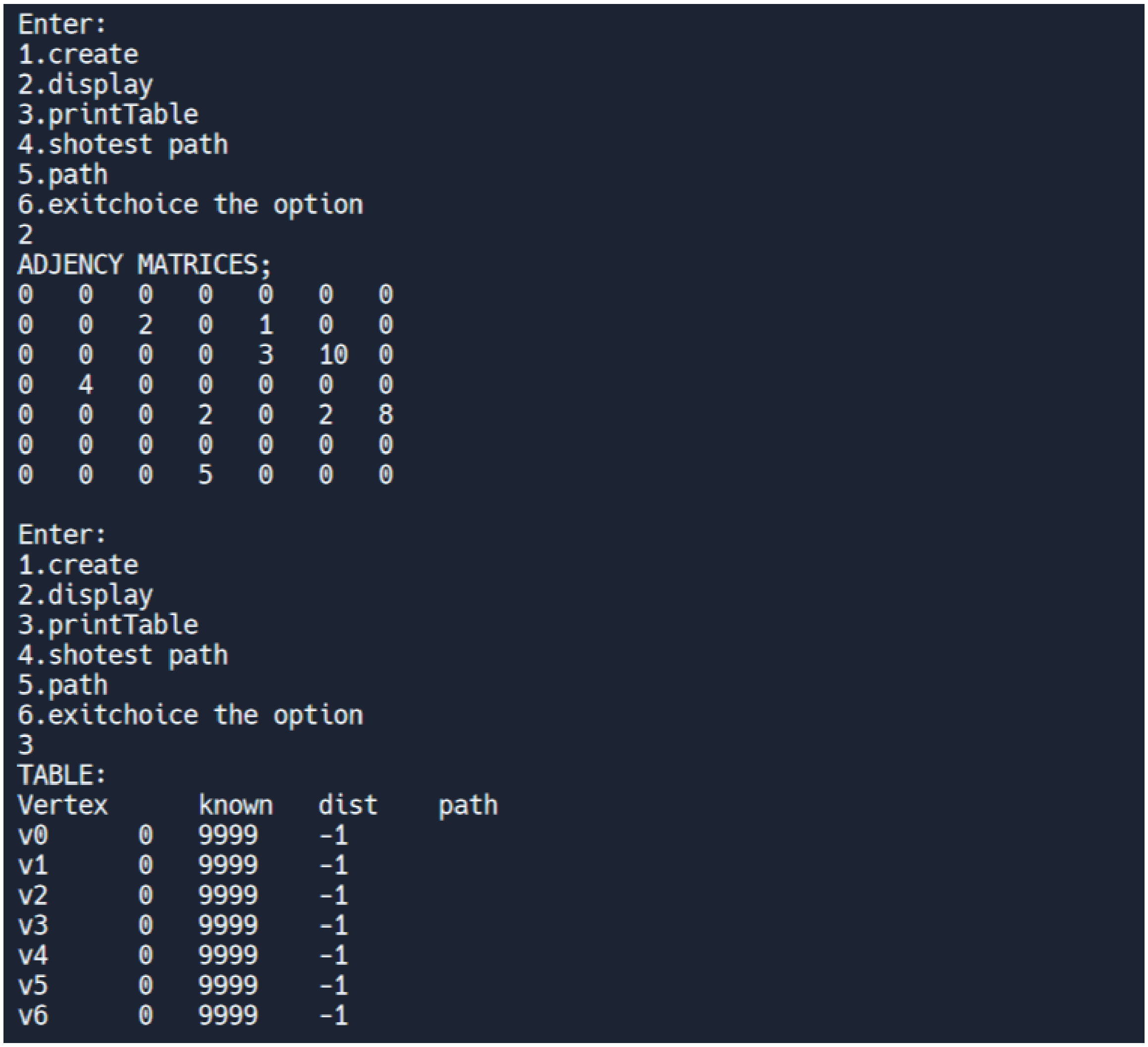
```
   for (int i = 0; i < edges; i++) {
    scanf("%d", &edge[i].from);
    scanf("%d", &edge[i].to);

    // scanf("%d",&edge[i].weight);
   }
   create(g, vertex, edges, edge);
   printf("\n");
   break;

  case 2:

   printf("ADJENCY MATRICES;\n");
   display(g);
   printf("\n");
   break;

  case 3:

   init(g);
   printf("TABLE:\n");
   printTable(g,vertex);
   printf("\n");
   break;
  case 4:

   printf("enter the starting vertex:\n");
   scanf("%d", &s);
   dijkstras(g, s,vertex);
   printf("\n");
   break;
  case 5:

   printf("Enter the destination: ");
   scanf("%d", &d);
   printf("PATH: \n");
   path(g, d);
   printf("\n");
   break;
  case 6:
   x = 0;
   printf("exit\n");
   break;
  }
```

```
 }
}
```

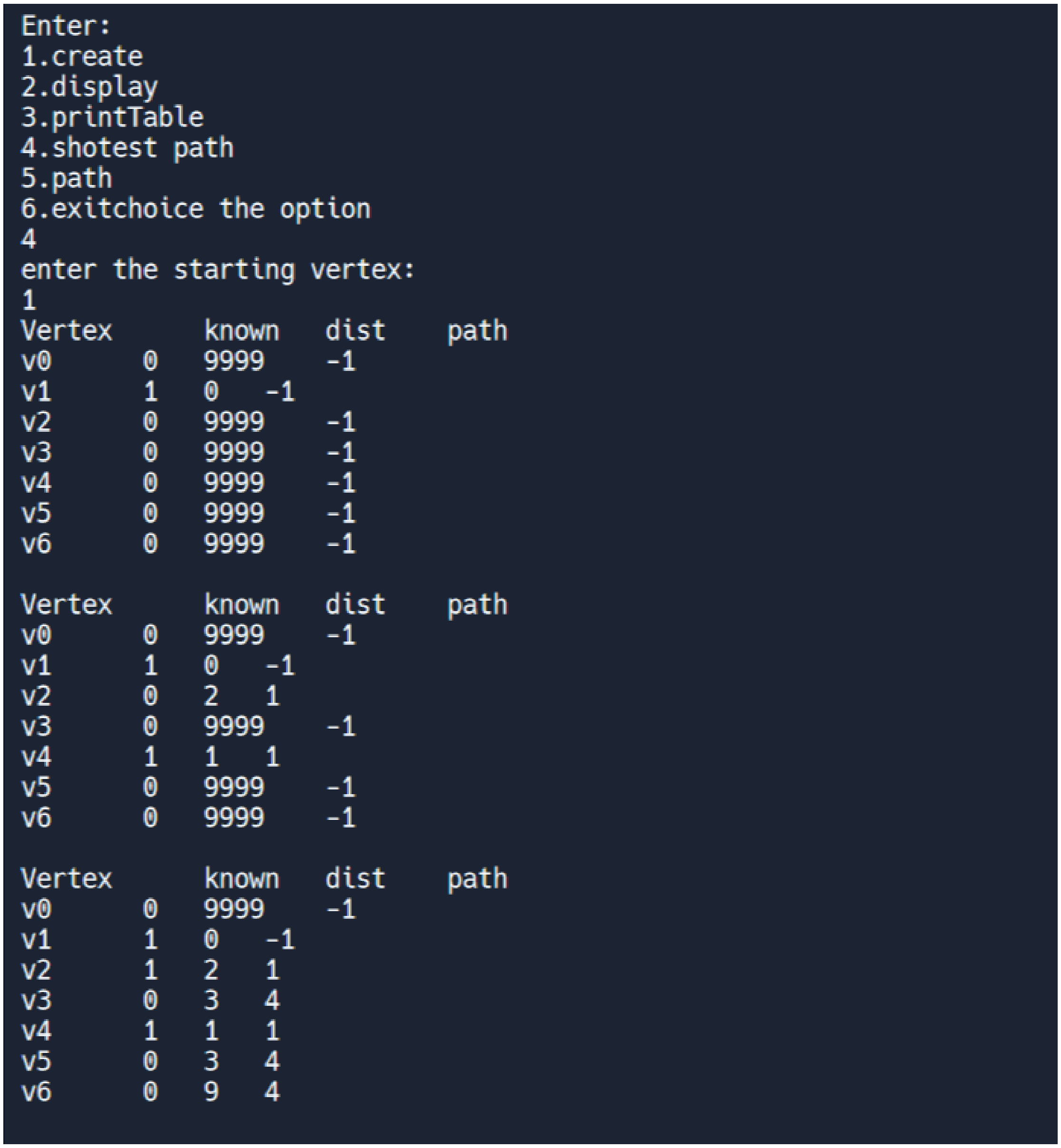
Output:



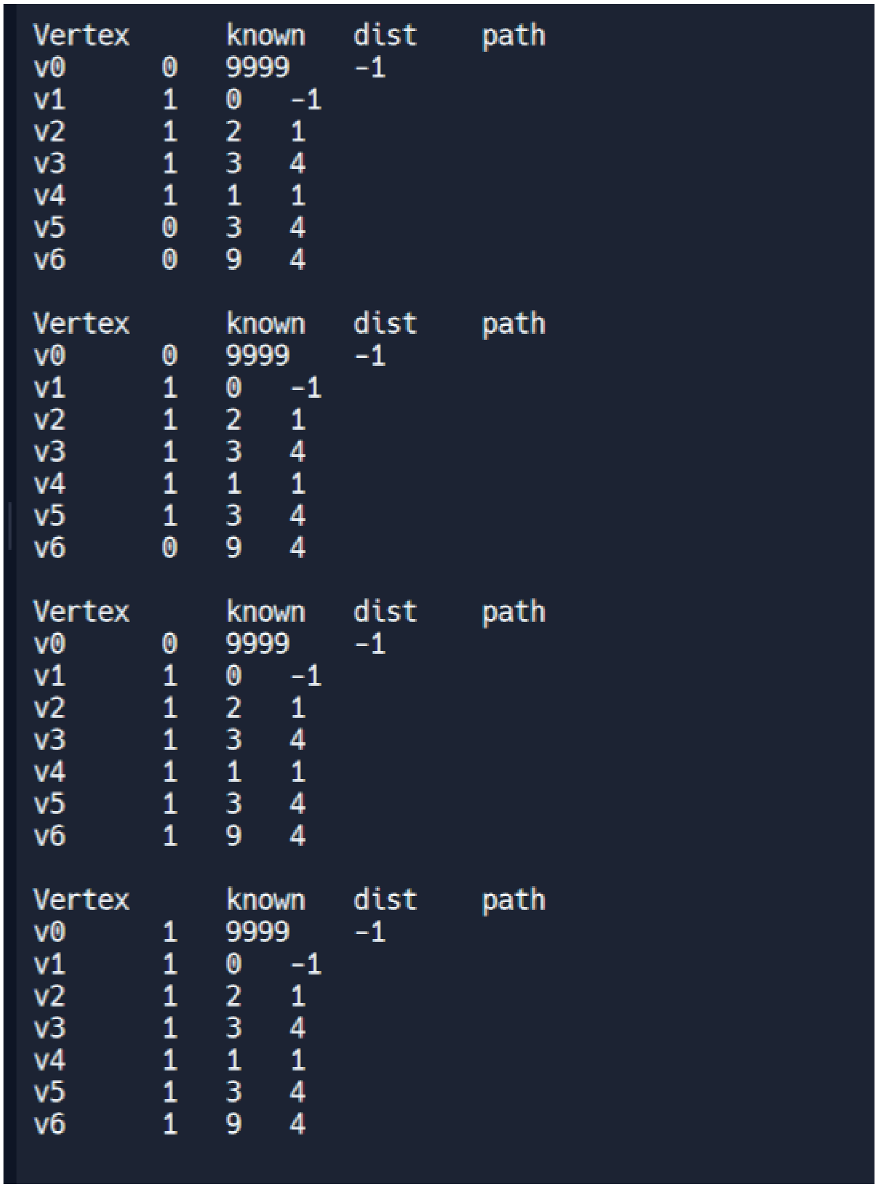
```
enter the number of vertex:7
enter the number of edges:12
Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
1
enter the edge pairs
1 2
2 5
5 7
7 6
6 3
3 1
1 4
2 4
4 5
4 7
4 6
4 3
enter the weight for(1,2);2
enter the weight for(2,5);10
enter the weight for(5,7);6
enter the weight for(7,6);1
enter the weight for(6,3);5
enter the weight for(3,1);4
enter the weight for(1,4);1
enter the weight for(2,4);3
enter the weight for(4,5);2
enter the weight for(4,7);4
enter the weight for(4,6);8
enter the weight for(4,3);2
```



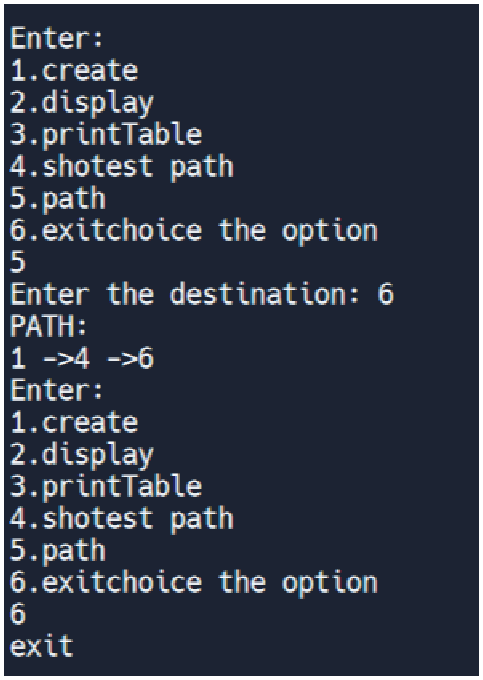
```
Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
2
ADJENCY MATRICES;
0   0   0   0   0   0   0
0   0   2   0   1   0   0
0   0   0   0   3   10  0
0   4   0   0   0   0   0
0   0   0   2   0   2   8
0   0   0   0   0   0   0
0   0   0   5   0   0   0

Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
3
TABLE:
Vertex      known   dist    path
v0      0   9999    -1
v1      0   9999    -1
v2      0   9999    -1
v3      0   9999    -1
v4      0   9999    -1
v5      0   9999    -1
v6      0   9999    -1
```



```
Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
4
enter the starting vertex:
1
Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      0   9999    -1
v3      0   9999    -1
v4      0   9999    -1
v5      0   9999    -1
v6      0   9999    -1

Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      0   2   1
v3      0   9999    -1
v4      1   1   1
v5      0   9999    -1
v6      0   9999    -1

Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      1   2   1
v3      0   3   4
v4      1   1   1
v5      0   3   4
v6      0   9   4
```



```
Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      1   2   1
v3      1   3   4
v4      1   1   1
v5      0   3   4
v6      0   9   4

Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      1   2   1
v3      1   3   4
v4      1   1   1
v5      1   3   4
v6      0   9   4

Vertex      known   dist    path
v0      0   9999    -1
v1      1   0   -1
v2      1   2   1
v3      1   3   4
v4      1   1   1
v5      1   3   4
v6      1   9   4

Vertex      known   dist    path
v0      1   9999    -1
v1      1   0   -1
v2      1   2   1
v3      1   3   4
v4      1   1   1
v5      1   3   4
v6      1   9   4
```



```
Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
5
Enter the destination: 6
PATH:
1 ->4 ->6
Enter:
1.create
2.display
3.printTable
4.shotest path
5.path
6.exitchoice the option
6
exit
```