

```

0001 // Display mode
0002 mode(0);
0003
0004 // Display warning for floating point exception
0005 ieee(1);
0006
0007 clc;
0008 siz = input("Enter the size of the matrix --> ");
0009 h = input("Enter the matrix --> ");
0010 s = input("Enter the source vertex --> ");
0011 d = input("Enter the destination vertex --> ");
0012
0013 for i =1:siz
0014     for j =1:siz
0015         if h(i,j) ==(-1) then
0016             h(i,j) = %inf;
0017         end;
0018     end;
0019 end;
0020
0021 matriz_costo = h;
0022 //function [sp, spcost] = dijkstra(matriz_costo, s, d)
0023
0024 n = siz;
0025 S(1:n) = 0; //s, vector, set of visited vectors
0026 dist(1:n) = %inf; // it stores the shortest distance between the source node and any other node;
0027 prev(1:n) = (n+1); // Previous node, informs about the best previous node known to reach each network
0028     node
0029 dist(s) = (0);
0030
0031 while (sum(S) ~= (n))
0032     candidate = [];
0033     for i =1:n
0034         if S(i)==0 then
0035             candidate = [candidate,dist(i)];
0036         else
0037             candidate = [candidate,%inf];
0038         end;
0039     end;
0040     [u_index,u] = min(candidate);
0041     S(u) = 1;
0042     for i = 1:n
0043         if (dist(u)+(matriz_costo(u,i))) < dist(i) then
0044             dist(i) = dist(u)+(matriz_costo(u,i));
0045             prev(i) = u;
0046         end;
0047     end;
0048 end;
0049 sp = [d];
0050 while (sp(1) ~= s )
0051     if (prev(sp(1)))<=n then
0052         sp = [prev(sp(1)) sp];
0053     else
0054         error;
0055     end;
0056 end;
0057 spcost = dist(d);
0058 // L.56: No simple equivalent, so mtlb_fprintf() is called.
0059 printf("\nThe least cost path from source node \"%d\" to destination node \"%d\" --> ",s,d);
0060 disp(sp);
0061 // L.58: No simple equivalent, so mtlb_fprintf() is called.
0062 printf("\nLeast cost is --> %g",spcost);
0063 // L.59: No simple equivalent, so mtlb_fprintf() is called.
0064 disp(" ----- PROCESSING COMPLETED -----");

```