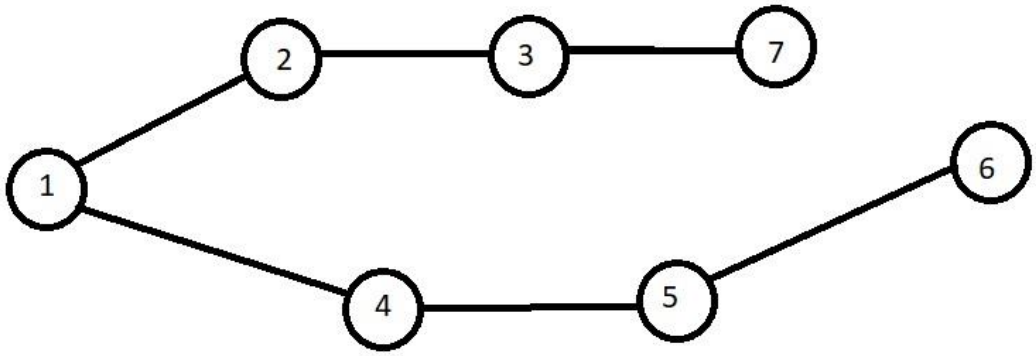
Aluno(a): Sthefany Sther Pequeno Alves;

RGM: 23535547;

Turma: 5A;

LISTA 1



Crie um algoritmo preferencialmente em linguagem C que procure um caminho entre
um vértice de origem e um vértice de destino.
O usuário entra com os vértices de origem e destino. Portanto pode ser escolhido qualquer um.
Entregar de preferência o algoritmo na sua completude.
Realizar os Testes abaixo:

Teste nº	origem	destino
1	1	2
2	1	3
3	1	4
4	1	6
5	2	4
6	3	6
7	5	3

CODIFICAÇÃO:

```
#include <stdio.h>

#include <locale.h>


#define qtdVertices 7

#define V 7
```

```
int mAdj [qtdVertices][qtdVertices];

void imprimirMatriz();

int caminho[100];
int mostrarCaminho = 0;

int VerOrigem;
int VerDestino;

void main (void){
	setlocale (LC_ALL, "Portuguese");
	printf ("\n----------Populando a matriz.----------");


	mAdj[0][1] = 2;
	mAdj[0][4] = 4;

	//vertice 2
	mAdj[1][2] = 3;
	mAdj[1][0] = 1;

	//vertice 3
	mAdj[2][1] = 2;
	mAdj[2][3] = 7;


	//vertice 4
	mAdj[3][0] = 1;
	mAdj[3][5] = 5;
```

```
	//vertice 5
	mAdj[4][4] = 4;
	mAdj[4][6] = 6;

	//vertice 6
	mAdj[5][5] = 5;

	//vertice 7
	mAdj[6][2] = 3;

	imprimirMatriz();
	printf("\n\nDigite o vértice origem:");
	scanf("%d", &VerOrigem);
	printf("\nDigite o vértice de destino:");
	scanf("%d", &VerDestino);
	EncontrarCaminho(VerOrigem, VerDestino, caminho);




}



void imprimirMatriz(){
	printf("\n\n----- Matriz Adjacência -----");
	int i, j;
	for(i = 0;i < qtdVertices;i++){
		printf("\n| ");
		for(j = 0;j < qtdVertices; j++){
			printf("\t%d",mAdj[i][j]);
		}
```

```
                printf(" |");
        }



}

void VoltarCaminho(){
        int r;
        for(r = 0; r < 100;r++){
                if(caminho[r] == 0){
                caminho[r-1] = 0;
                }
        }
}

// Imprime o caminho sendo pesquisado atualmente
void MostrarCaminho(int caminho[]){
        int x =0;
        if(caminho[x] == 0){printf("Não ha caminho a ser mostrado.");}
        else{
                printf(" %d",caminho[x]);
                x ++;
                if(caminho[x] == 0){}
                else{

                        do{

                        printf(" => %d",caminho[x]);
                        x = x+1;
                        }
                        while(caminho[x] != 0);
                }
```

```
        }
}

// imprimi a matriz
void imprimir(){
        printf("\n");
        int i,j,k,l;
        printf(" ");
        for(k=0;k<V;k++){
                printf("%d ",k+1);
        }
        printf("\n ");
        for(l=0;l<V;l++){
                printf("____");
        }
        printf("\n");
        for(i=0;i < V;i++){
                printf("%d | ", i+1);
                for(j=0;j < V; j++){
                        printf("%d ",mAdj[i][j]);
                }
                printf("\n");
        }
}

// Acredito que mostra o(s) vertice(s) vizinho(s) ao selecionado
void vizinho(int v){
        printf("\nO(s) vizinho(s) do vertice %d e(são): ", v);
        v = v - 1;

        int j;
        for(j = 0; j < V; j++){
```

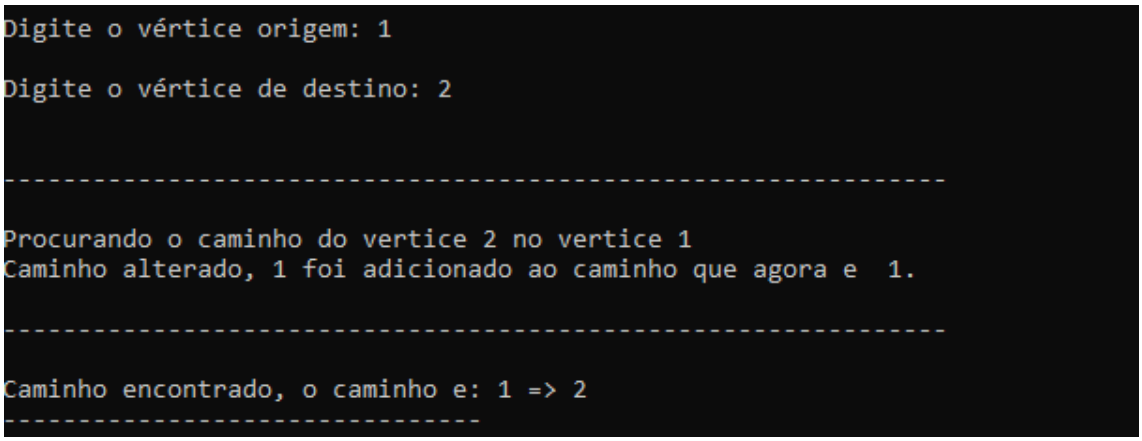
```
                if(mAdj[v][j] == 1){
                printf("%d, ", j+1 );
                }
        }
}

// Encontra o caminho de qualquer vertice para qualquer vertice
int EncontrarCaminho(int vOrigem, int vDestino, int caminho[]){
        printf("\n\n----------------------------------------------------------------\n");
        if(vDestino != vOrigem){
        printf("\nProcurando o caminho do vertice %d no vertice %d", vDestino, vOrigem);
        }

        vDestino --;
        vOrigem --;
        int estaNoCaminho = 0;
        int i,x,c,ca;

        //Adiociona o numero origem ao final da lista ciminho
        for(ca = 0;ca<100;ca++){
        if(caminho[ca] == 0){
        caminho[ca] = vOrigem+1;
        if(vDestino != vOrigem){
        printf("\nCaminho alterado, ");
        printf("%d foi adicionado ao caminho que agora e ", vOrigem+1);
        MostrarCaminho(caminho);
        printf(".");
        }
        break;
        }
        }
```

```
// Confere se o número atualmente sendo analisado e o desejado

for(x = 0; x<100; x++){
if(vDestino+1 == caminho[x]){
printf("\nCaminho encontrado, o caminho e:");
MostrarCaminho(caminho);
mostrarCaminho = 1;
return 1;
}
}

// Define se o vertice que esta atualmente em pesquisa ja esta no caminho passado
if(mostrarCaminho == 0){
	// printf("\n\nMostrar Caminho: %d\n\n", mostrarCaminho);
	for(i = 0; i < V; i++){
	// printf("\nAtualmente analisando %d na coluna de %d ", mAdj[vOrigem][i],vOrigem);
	estaNoCaminho = 0;
	for(c = 0; c < V;c++){
	if(mAdj[vOrigem][i] == caminho[c]){
	estaNoCaminho = 1;
	}
	}

	// Testa a fileira do vertice em quesão, se não e um zero e não estiver no caminho passado, toma como alvo
	if(mAdj[vOrigem][i] != 0 && estaNoCaminho == 0){
	// printf("\nAchou um novo vertice, o %d", mAdj[vOrigem][i]);
	if(EncontrarCaminho(mAdj[vOrigem][i],vDestino+1,caminho) == 1){
	return 1;
	}
	else{
	VoltarCaminho();
```

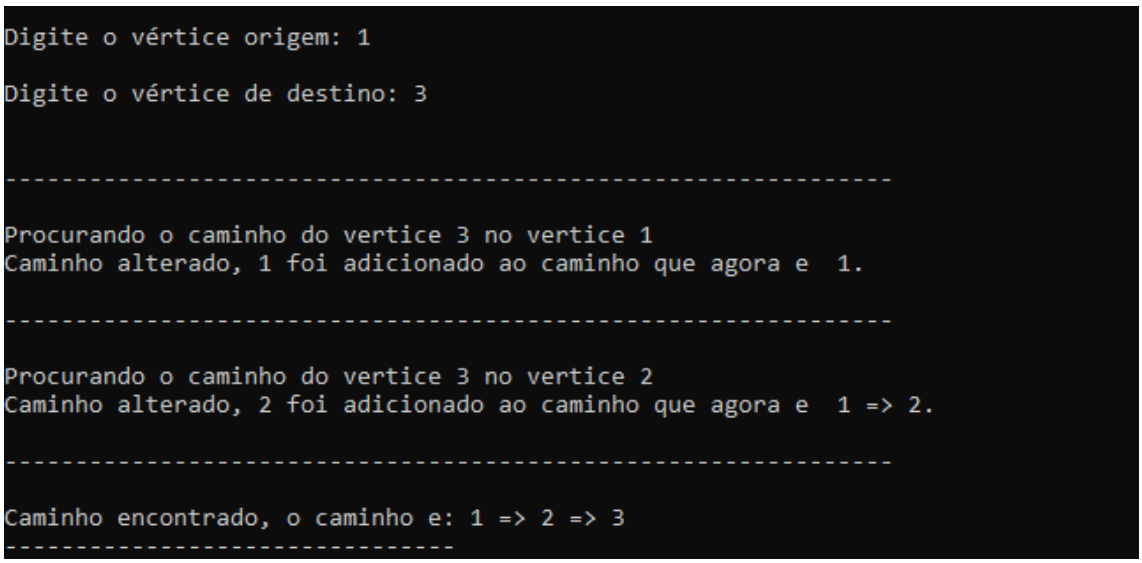
```
                // printf("\nVoltou um Vertice");


                }
                }
                }
                return 0;
        }
}
```

Os códigos seguindo os caminhos propostos.

1 ao 2



1 ao 3



1 ao 4

```
Digite o vértice origem: 1

Digite o vértice de destino: 4


---------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 1
Caminho alterado, 1 foi adicionado ao caminho que agora e  1.

---------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 2
Caminho alterado, 2 foi adicionado ao caminho que agora e  1 => 2.

---------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 3
Caminho alterado, 3 foi adicionado ao caminho que agora e  1 => 2 => 3.

---------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 7
Caminho alterado, 7 foi adicionado ao caminho que agora e  1 => 2 => 3 => 7.

---------------------------------------------------------------

Caminho encontrado, o caminho e: 1 => 4
---------------------------------
```

1 ao 6

```
Digite o vértice origem: 1

Digite o vértice de destino: 6


---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 1
Caminho alterado, 1 foi adicionado ao caminho que agora e  1.

---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 2
Caminho alterado, 2 foi adicionado ao caminho que agora e  1 => 2.

---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 3
Caminho alterado, 3 foi adicionado ao caminho que agora e  1 => 2 => 3.

---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 7
Caminho alterado, 7 foi adicionado ao caminho que agora e  1 => 2 => 3 => 7.

---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 4
Caminho alterado, 4 foi adicionado ao caminho que agora e  1 => 4.

---------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 5
Caminho alterado, 5 foi adicionado ao caminho que agora e  1 => 4 => 5.

---------------------------------------------------------------

Caminho encontrado, o caminho e: 1 => 4 => 5 => 6
---------------------------------
```

2 ao 4

```
Digite o vértice origem: 2

Digite o vértice de destino: 4


------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 2
Caminho alterado, 2 foi adicionado ao caminho que agora e  2.

------------------------------------------------------------

Procurando o caminho do vertice 4 no vertice 1
Caminho alterado, 1 foi adicionado ao caminho que agora e  2 => 1.

------------------------------------------------------------

Caminho encontrado, o caminho e: 2 => 1 => 4
--------------------------------
```

3 ao 6

```
Digite o vértice origem: 3

Digite o vértice de destino: 6


------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 3
Caminho alterado, 3 foi adicionado ao caminho que agora e  3.

------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 2
Caminho alterado, 2 foi adicionado ao caminho que agora e  3 => 2.

------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 1
Caminho alterado, 1 foi adicionado ao caminho que agora e  3 => 2 => 1.

------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 4
Caminho alterado, 4 foi adicionado ao caminho que agora e  3 => 2 => 1 => 4.

------------------------------------------------------------

Procurando o caminho do vertice 6 no vertice 5
Caminho alterado, 5 foi adicionado ao caminho que agora e  3 => 2 => 1 => 4 => 5.

------------------------------------------------------------

Caminho encontrado, o caminho e: 3 => 2 => 1 => 4 => 5 => 6
--------------------------------
```

5 ao 3

```
Digite o vértice origem: 5

Digite o vértice de destino: 3


---------------------------------------------------------------

Procurando o caminho do vertice 3 no vertice 5
Caminho alterado, 5 foi adicionado ao caminho que agora e  5.

---------------------------------------------------------------

Procurando o caminho do vertice 3 no vertice 4
Caminho alterado, 4 foi adicionado ao caminho que agora e  5 => 4.

---------------------------------------------------------------

Procurando o caminho do vertice 3 no vertice 1
Caminho alterado, 1 foi adicionado ao caminho que agora e  5 => 4 => 1.

---------------------------------------------------------------

Procurando o caminho do vertice 3 no vertice 2
Caminho alterado, 2 foi adicionado ao caminho que agora e  5 => 4 => 1 => 2.

---------------------------------------------------------------

Caminho encontrado, o caminho e: 5 => 4 => 1 => 2 => 3
```