Notas:

1. Para representar un Grafo utilicé una Lista generalizada.
2. Modifique las funciones incluidas en Grafos.
3. De acuerdo a la implementación la función InicializaValorVertice pierde sentido.
4. Grafos funciona perfectamente al igual que las funciones.
5. La función camino Hamiltoniano funciona, sin embargo, tiene defectos además la función camino Euleriano funciona, sin embargo, tiene defectos.

Instrucciones

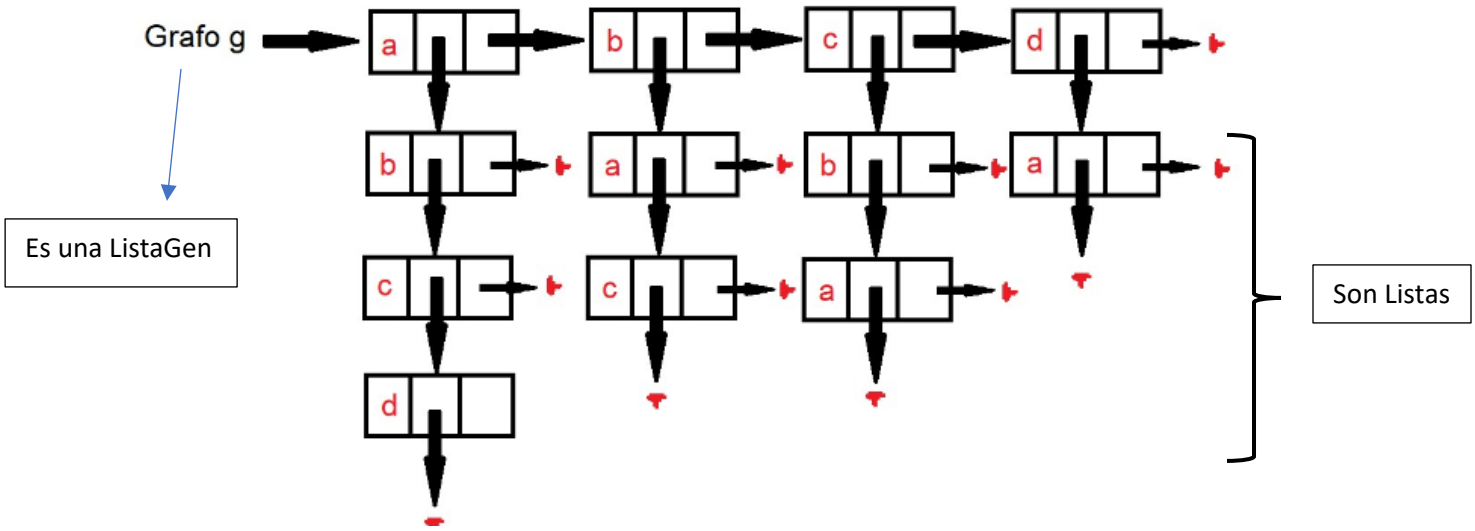
Menu.c Permite crear un Grafo y manipularlo mediante un menú, sin embargo, no puede determinar su Lista Hamiltoniana.

p.c Permite crear un grafo y manipularlo, sin embargo, no tiene un menú, por defecto crea un grafo y muestra la lista hamiltoniana, al ejecutarlo deberá ingresar el carácter de cada arista.

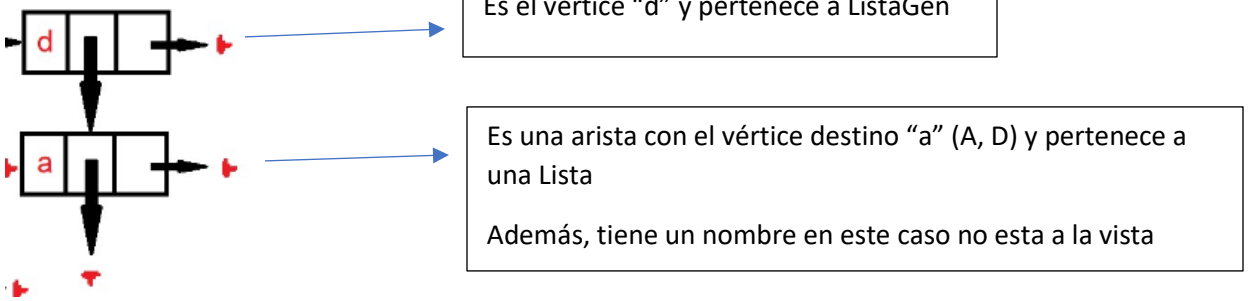
pEuleriano.c Permite crear un grafo y manipularlo, sin embargo, no tiene un menú, por defecto crea un grafo y muestra la lista Euleriana, al ejecutarlo deberá ingresar el carácter de cada arista.

Para representar la estructura de datos grafo utilice una lista generalizada que contiene el nombre del vértice y una lista, la cual contiene el nombre de la arista y el vértice final o destino.

Es decir, una ListaGen contiene el nombre del vértice y la lista de sus vecinos.



Cada nodo de una lista contiene el nombre de la arista y el vértice destino.



Nótese que ListaGen es diferente de Lista, ya que ListaGen es una lista de vértices por el contrario Lista es una lista de aristas que pertenecen a un vértice.

```
typedef struct Nodo{

    ElemArista dt;
    struct Nodo *sig;

}*Lista;

typedef struct NodoListaGen{

    Elem dato;
    Lista l;
    struct NodoListaGen *sig;

}*ListaGen;
```

Explicación Elem.h

La estructura ElemArista es un tipo de dato que contiene el nombre de la Arista y el vértice destino.

```
typedef struct {

    Arista N;
    Vertice y;

}ElemArista;
```

ImpElemArista(ElemArista)->; Esta función imprime un elemento de tipo ElemArista;

ImpElemHamilt(ElemArista)->; Esta función imprime un elemento de tipo ElemArista sin embargo modifica el contenido para que imprima el vértice inicio y el vértice destino.

EscribeElemArista(ElemArista) -> Arista; Esta función recoge el nombre de una arista.

VerticesIguales(Vertice, Vertice) -> bool; Esta función compara dos tipos de datos Vértice.

EsIgualArista(ElemArista, ElemArista) -> bool; Esta función compara dos tipos de datos Arista.

Explicación ElemVertice.h

La estructura ElemVertice es un tipo de dato que contiene el Vértice y una estructura Lista.

```
typedef struct{

    Vertice x;
    Lista l;

}ElemVertice;
```

HazElemVertice(Elem, Lista) -> ElemVertice; Esta función recoge un Elem y una lista donde los guarda en una estructura de tipo ElemVertice

Explicacion de ListaGen.h

Note que ListaGen es diferente de Lista.

Funciones para ListaGen.

VaciaG() -> ListaGen; Regresa una ListaGen vacia.

EsVaciaG(ListaGen lg) -> bool; Pregunta si una ListaGen es vacia.

ConsG(Elem e, ListaGen lg) -> ListaGen; Construye una ListaGen con un Elem sin embargo define la Lista como vacia.

EditG(Elem e, Lista li, ListaGen lg) -> ListaGen; Construye una ListaGen con un Elem sin embargo la Lista no es vacia.

CabezaG(ListaGen lg) -> Elem; Regresa el último elemento ingresado en la ListaGen.

ListaG(ListaGen lg) -> Lista; Regresa la Lista Asociada al ultimo elemento ingresado en la ListaGen.

RestoG(ListaGen lg) -> ListaGen; Regresa una ListaGen retirando el último elemento y la Lista asociada a este.

EliminaG(Elem e, ListaGen lg) -> ListaGen; Elimina un elemento y la Lista asociada a este sin importar cuando se ingresó.

ImpGen(ListaGen lg) -> ; Imprime los elementos de la ListaGen sin embargo no imprime las listas asociadas a estos.

EstaEnG(Elem e,ListaGen lg) -> bool; Pregunta si un elemento se encuentra en una ListaGen.

Funciones para Lista

VaciaL() -> Lista; Regresa una Lista vacía.

EsVaciaL(Lista l) -> bool; Pregunta si una Lista es vacía.

ConsL(ElemArista e, Lista l) -> Lista ; Construye una Lista con una estructura ElemArista.

CabezaL(Lista l) -> ElemArista ; Regresa el ultimo ElemArista ingresado en la Lista.

RestoL(Lista l) -> Lista ; Regresa una Lista sin el ultimo ElemArista ingresado.

EliminaL(ElemArista e, Lista l) -> Lista ; Elimina un ElemArista ingresado en una Lista.

ConsultaL(Elem e, Lista l) -> ElemArista ; Regresa un ElemArista contenido en una Lista.

EstaEnL(Elem e,Lista l) -> bool; Pregunta si un ElemArista está dentro de una Lista.

NumElemList(Lista l) -> int; Regresa la cantidad de elementos de una Lista.

ImpListaL(Lista l) -> ; Imprime una Lista con el formato (Arista , Vértice).

ImpListHamilt(Lista l) -> ; Imprime una Lista con el formato (Vértice, Vértice).

Funciones complementarias para el uso de Grafos

ImpComp(ListaGen lg) -> ; Imprime un Elem junto con la Lista asociada a este con el formato "Elem[Lista]"

LlenaLista(Elem e, ElemArista v, ListaGen lg) -> ListaGen ; Ingresa un ElemArista en una Lista asociada a un Elem perteneciente a una ListaGen.

EliminaLista(Elem e, ElemArista v, ListaGen lg) -> ListaGen; Elimina un ElemArista en una Lista asociada a un Elem perteneciente a una ListaGen.

EliminaAristaVertice(Elem e, ListaGen lg) -> ListaGen; Elimina todas los ElemAristas que están relacionados a un Elem dentro de una ListaGen.

Es decir, si eliminas un Elem=Vértice se encargará de eliminar todas las aristas que tienen como destino el Elem=Vértice dentro de una ListaGen.

EstaArista(Elem e1, Elem e2, ListaGen g) -> bool; Pregunta si un ElemArista tiene como vértice destino un Elem, dentro de un Elem.

Es decir, pregunta si existe una arista que conecte el vértice e1 con el vértice e2 (e1, e2).

La explicación para Grafos.h es trivial puesto la mayoría de las funciones siguen el formato establecido.

DeteH(Grafo g, Lista l, ElemArista e) -> Lista ; Complemento para determinar una Lista Hamiltoniana de un Grafo.

ListaHamiltoniana(Grafo g) -> Lista ; Obtiene una Lista Hamiltoniana de un Grafo .

EsEuleriano(Grafo g,int i) –> bool ; Pregunta si un grafo es euleriano.

Notas extensas:

No implemente la función Camino Euleriano, sin embargo, implemente la función para determinar si un Grafo es Euleriano.

La función ListaHamiltoniana tiene algunos defectos puesto que solo funciona con ciertos casos ya que el orden en el cual están las aristas de cada vértice influye, ha sido probada en varios Grafos y funciona, esta puede mejorarse sin embargo modificaría algunas funciones de ListaGen, dentro del programa añadí dos grafos únicamente ingrese los nombres de las aristas(caracteres).

La función ListaEuleriana tiene algunos defectos puesto que solo funciona con ciertos casos ya que el orden en el cual están las aristas de cada vértice influye, ha sido probada en varios Grafos y

funciona, dentro del programa añadí dos grafos únicamente ingresé los nombres de las aristas(caracteres) y desplegara la lista Euleriana.

Formato de impresión camino euleriano y hamiltoniano.

[Vértice: DESTINO, Vértice ORIGEN]

Formato de impresión Grafo

Vértice [Arista: Nombre, Vértice: Destino], Vértice [Arista: Nombre, Vértice: Destino]