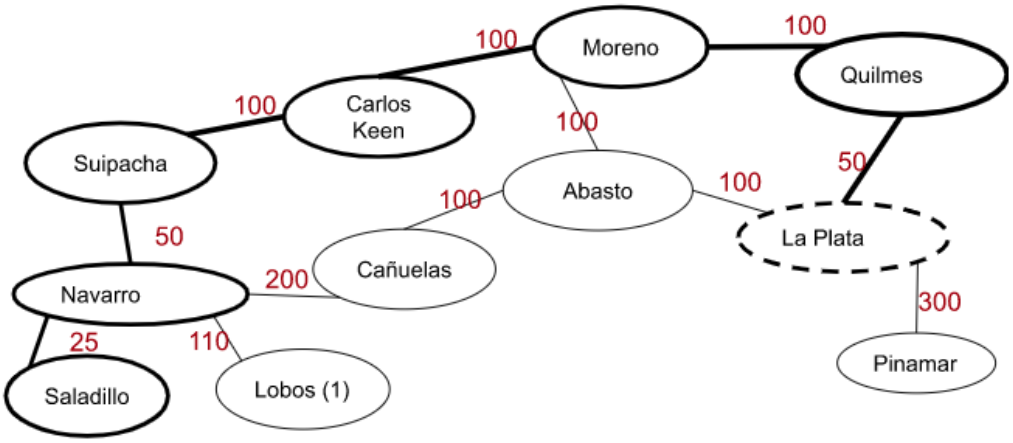
Implemente la clase **Parcial**, y el método:

???? resolver(Grafo<????> ciudades, String origen, int maxMonto)

Se quiere realizar un viaje desde una ciudad origen, recorriendo la mayor cantidad de localidades de la provincia. Sin embargo se tiene un presupuesto limitado para gastar en los peajes de las distintas rutas (“maxMonto”). Su algoritmo debería devolver el camino que cumple con la condición mencionada y el monto sobrante del gasto en peajes. Si hubieran 2 caminos que permiten recorrer la misma cantidad de ciudades, debe devolver el que permita el mayor ahorro.

En el siguiente ejemplo, partiendo de **La Plata** con un presupuesto máximo de $500, el mejor resultado sería el camino que pasa por: La Plata, Quilmes, Moreno, Carlos Keen, Suipacha, Navarro, Saladillo, y sobran $75 (diferencia de los $500 originales y los $425 del camino).

Si bien hay otro camino con el mismo número de ciudades (La Plata, Abasto, Moreno, Carlos Keen, Suipacha, Navarro, Saladillo), el ahorro obtenido es menor ($25 resultante de la diferencia de los $500 originales y los $475 del camino).



Nota:

- Complete en la firma del método los tipos de datos ejemplificados con signo de interrogación
- Debe verificar la existencia de la ciudad origen
- No puede pasar más de 1 vez por la misma ciudad
- En caso de existir 2 resultados que cumplen el criterio dado, puede retornar cualquiera
- En caso de no existir un camino posible, debe devolver la lista vacía.
- Use los métodos de Grafo y Listas vistos en clase.

Posible solución:

```
public class Parcial {

    public Resultado resolver(Grafo<String> ciudades, String origen, int montoMax) {

        Resultado resultado = new Resultado();
        resultado.setMontoSobrante(montoMax);
        if (ciudades != null && !ciudades.esVacio()) {
            ListaGenerica<String> camino = new ListaGenericaEnlazada<String>();
            boolean[] visitados = new boolean[ciudades.listaDeVertices().tamanio()];
            ListaGenerica<Vertice<String>> vertices = ciudades.listaDeVertices();
            vertices.comenzar();
            Vertice<String> vInicial = null;
            while (!vertices.fin() && vInicial == null) {
                Vertice<String> vertice = vertices.proximo();
                if (vertice.dato().equals(origen)) {
                    vInicial = vertice;
                }
            }
            if (vInicial != null) {
                dfs(vInicial, ciudades, visitados, resultado, camino, montoMax);
            }
        }
        return resultado;

    }
```

```
private void dfs(Vertice<String> vInicial, Grafo<String> ciudades, boolean[] visitados, Resultado resultado,
        ListaGenerica<String> camino, int montoMax) {
    visitados[vInicial.posicion()] = true;
    camino.agregarFinal(vInicial.dato());

    if ((camino.tamanio() > resultado.getCamino().tamanio())
            || (camino.tamanio() == resultado.getCamino().tamanio() && resultado.getMontoSobrante() < montoMax)) {
        resultado.setCamino(camino.clonar());
        resultado.setMontoSobrante(montoMax);
    }

    ListaGenerica<Arista<String>> adyacentes = ciudades.listaDeAdyacentes(vInicial);
    adyacentes.comenzar();
    while (!adyacentes.fin()) {
        Arista<String> arista = adyacentes.proximo();
        int posicion = arista.verticeDestino().posicion();
        if (montoMax - arista.peso() >= 0 && !visitados[posicion]) {
            dfs(arista.verticeDestino(), ciudades, visitados, resultado, camino, montoMax - arista.peso());
        }
    }

    visitados[vInicial.posicion()] = false;
    camino.eliminarEn(camino.tamanio()-1);

}
```

```java
public class Resultado {

    private ListaGenerica<String> camino;
    private int montoSobrante;

    public Resultado() {
        camino = new ListaGenericaEnlazada<String>();
        montoSobrante = 0;
    }
    public ListaGenerica<String> getCamino() {
        return camino;
    }
    public void setCamino(ListaGenerica<String> camino) {
        this.camino = camino;
    }
    public int getMontoSobrante() {
        return montoSobrante;
    }
    public void setMontoSobrante(int montoSobrante) {
        this.montoSobrante = montoSobrante;
    }


}
```