6)

```
public class VisitaOslo {
  public ListaGenerica<string> paseoEnBici (Grafo<string> lugares, String
  destino, int maxTiempo, ListaGenerica<string> lugaresRestringidos){
    ListaGenerica<string> c = new ListaEnlazadaGenerica<string>();
    if(!lugares.esVacio()){
      Vertice<string> V = buscar("Ayuntamiento", lugares);
      if(V!=null){
        boolean[] marcas = new boolean[lugares.listaDeVertices().
        tamanio()+1];
        marcar(marcas, lugaresRegistringidos, lugares);
        dfs(c, marcas, V.posicion(), maxTiempo, destino, lugares);
      }
    }
    return c;
  }
  private buscar(String a, Grafo g){
    Vertice<string> V = null;
    ListaGenerica<string> l = g.listaDeVertices();
    l.comenzar();
    while(!l.Fin() && V==null){
      Vertice<string> Ve = l.proximo();
      if(Ve.dato().equals(a)){ V = Ve;}
    }
    return V;
  }
}
```

```
private Void marcar(boolean[] m, ListaGenerica<String> l, Grafo g);
  l.comenzar();
  while(!l.fin()){
    m[buscar(l.proximo().dato(), g).posicion()]=True;
  }
}

private boolean dfs(Lista Generica<string> c, boolean[] m, int i, int
tiempo, String d, Grafo g){
  Vertice<String> V = g.vertice(i);
  c.agregarAlFinal(V.dato());
  m[i] = True; encontre = False;
  if(V.dato.equals(d)){
    encontre=True;
  }else{
    ListaGenerica<Arista<string>>a= g.listaDeAdyacentes(V);
    a.comenzar();
    while(!a.fin() && !encontre){
      Arista<String> ar = a.proximo();
      int j = ar.VerticeDestino().posicion();
      if(!m[j] && tiempo - ar.peso() > 0){
        encontre = dfs(c, m, j, tiempo-ar.peso(), d, g);
      }
    }
  }
  if(!encontre){c.eliminar(c.tamanio());}
  return encontre;
}
```