Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2017)

Solución Control 2

Nombre: _____

Nº alumno: _____

Pregunta 1

Indique el output del siguiente código:

```
class Persona:
    def __init__(self, nombre):
        self.nombre = nombre
    def __repr__(self):
        return self.nombre
class Nodo:
    def __init__(self, valor):
        self.valor = valor
    def __repr__(self):
        return repr(self.valor)
class Grafo:
    def __init__(self, lista_adyacencia=None):
        self.lista_adyacencia = dict() if lista_adyacencia \
        is None else lista_adyacencia

def bfs(graph, start):
    visited, queue = list(), [start]
    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.append(vertex)
            for v in graph[vertex]:
                if v not in visited:
                    queue.append(v)
    print(visited)

def dfs(graph, start):
    visited, stack = list(), [start]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
```

```
            visited.append(vertex)
            for v in graph[vertex]:
                if v not in visited:
                    stack.append(v)
    print(visited)

bamavrakis = Nodo(Persona("Bastian"))
hernan4444 = Nodo(Persona("Hernan"))
diflores = Nodo(Persona("Daniela"))
hanavarrete = Nodo(Persona("Hugo"))
vincevalence = Nodo(Persona("Vicente"))
fnquinteros = Nodo(Persona("Felipe"))

amistades = {
    bamavrakis: set([hernan4444, diflores, hanavarrete, fnquinteros]),
    hernan4444: set([hanavarrete, fnquinteros]),
    diflores: set([hernan4444, vincevalence]),
    vincevalence: set([diflores, fnquinteros]),
    hanavarrete: set([hernan4444, diflores, vincevalence]),
    fnquinteros: set([diflores, bamavrakis]),
}
grafo = Grafo(amistades)

bfs(amistades, hanavarrete)
dfs(amistades, hanavarrete)
```

Solución:

```
[Hugo, Daniela, Vicente, Hernan, Felipe, Bastian]
[Hugo, Hernan, Felipe, Bastian, Daniela, Vicente]
```

Pregunta 2

Describa un procedimiento recursivo que permita obtener la profundidad de un árbol binario, *i.e.*, el camino más largo desde el nodo raíz hasta un nodo hoja. Puede entregar su respuesta como una descripción textual, código en Python o pseudocódigo.

Solución: Dado un árbol binario t, definimos la función $h(t)$, que calcula la profundidad del árbol t, de la siguiente manera:

$$h(t) = \begin{cases} -1 & \text{si } t \text{ es vacío,} \\ 1 + \max(h(subArbolIzq(t)), h(subArbolDer(t))) & \text{en otro caso.} \end{cases}$$