v=[[0,100,0,0,0,0,0],[0,0,0,200,0,800,0],[25,30,0,0,0,0,0],[0,0,50,0,50,0,0],[0,25,0,0,0,0,0],[0,0,0,0,50,0,40],[0,0,0,80,35,0,0]]

for f in range(7):

for c in range(7):

print (v[f][c],end=" ")

print()

n=[[0,1,0,0,0,0,0],[0,0,0,1,0,1,0],[1,1,0,0,0,0,0],[0,0,1,0,1,0,0],[0,1,0,0,0,0,0],[0,0,0,0,1,0,1],[0,0,0,1,1,0,0]]

for f in range(7):

for c in range(7):

print (n[f][c],end=" ")

print()

class Nodo:

def \_\_init\_\_(self, numero=None, nombre=None, apellido=None, telefono=None, direccion=None,

izquierda=None, derecha=None):

self.numero = numero

self.nombre = nombre

self.apellido = apellido

self.telefono = telefono

self.direccion = direccion

self.izquierda = izquierda

self.derecha = derecha

def \_\_str\_\_(self):

return "Nodo-->%s Nombre: %s Apellido: %s Telefono: %s Direccion: %s" % (self.numero, self.nombre, self.apellido, self.telefono, self.direccion)

class AB:

def \_\_init\_\_(self):

self.raiz = None

def agregar(self, elemento):

if self.raiz is None:

self.raiz = elemento

else:

auxiliar = self.raiz

padre = None

while auxiliar is not None:

padre = auxiliar

if int(elemento.numero) >= int(auxiliar.numero):

auxiliar = auxiliar.derecha

else:

auxiliar = auxiliar.izquierda

if int(elemento.numero) >= int(padre.numero):

padre.derecha = elemento

else:

padre.izquierda = elemento

def preorden(self, elemento):

if elemento is not None:

print(elemento)

self.preorden(elemento.izquierda)

self.preorden(elemento.derecha)

def postorden(self, elemento):

if elemento is not None:

self.postorden(elemento.izquierda)

self.postorden(elemento.derecha)

print(elemento)

def inorden(self, elemento):

if elemento is not None:

self.inorden(elemento.izquierda)

print(elemento)

self.inorden(elemento.derecha)

# Agrege esta funcion adicional llamada buscar para imprimir el contenido de un nodo a partir del numero de nodo

def buscar(self, elemento, nodonumero):

if elemento is not None:

if nodonumero == elemento.numero:

return elemento

else:

if nodonumero < elemento.numero:

return self.buscar(elemento.izquierda, nodonumero)

else:

return self.buscar(elemento.derecha, nodonumero)

def obtenerraiz(self):

return self.raiz

if \_\_name\_\_ == "\_\_main\_\_":

ab = AB()

while True:

print(" Menu ")

print("1.Agregar")

print("2. Organizar por preorden")

print("3. Organizar por postorden")

print("4. Organizar por inorden")

print("5. Buscar nodo")

num = input("Elegir la opcion:")

if num == "1":

n = input("Ingrese un numero: ")

nom = input("Ingrese un nombre: ")

ap = input("Ingrese un apellido: ")

tel = input("Ingrese el telefono: ")

dire = input("Ingrese direccion: ")

nodo = Nodo(n, nom, ap, tel, dire)

ab.agregar(nodo)

elif num == "2":

print("...Imprimiendo en preorden...")

ab.preorden(ab.obtenerraiz())

elif num == "3":

print("...Imprimiendo en postorden...")

ab.postorden(ab.obtenerraiz())

elif num == "4":

print("...Imprimiendo por inorden")

ab.inorden(ab.obtenerraiz())