| **Nama : Maulana ikhlasul ahnaf**  **NIM : 064002100016** | **Algoritma dan Pemrograman Dasar** | **Modul 11**  **Nama Dosen:**   1. Abdul Rochman 2. Anung B. Ariwibowo |
| --- | --- | --- |
| **Hari/Tanggal:**  Senin, 13 Juni 2022 | **Nama Aslab:**   1. Grace Ester A. (064001900014) 2. Rifdah Amelia (064001900019) |
|

**MODUL 11 : AVL TREE 2**

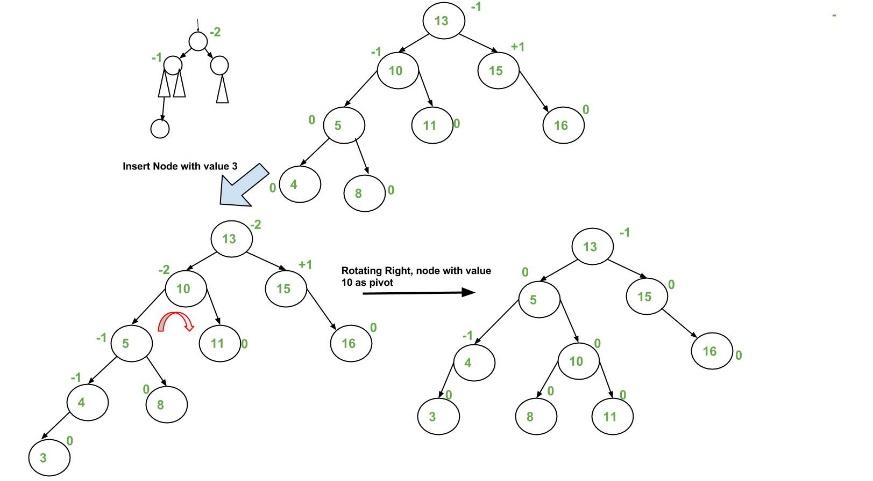
**Deskripsi Modul :** Memahami dan menerapkan ilmu struktur data dan algoritma untuk menyelesaikan masalah yang disajikan dengan menggunakan program berbasis bahasa Python.

| **No.** | **Elemen Kompetensi** | **Indikator Kinerja** | **Halaman** |
| --- | --- | --- | --- |
| 1. | Mampu memahami dan mengimplementasikan AVL Tree pada Python | Membuat dan memahami sebuah program yang menerapkan AVL Tree. |  |

**TEORI SINGKAT**

AVL Tree adalah Binary Search Tree yang memiliki perbedaan tinggi / level antara sub-tree kanan dan kirinya maksimal 1. AVL Tree digunakan untuk menyeimbangkan Binary Search Tree. Dengan menggunakan AVL Tree waktu pencarian dan bentuk tree yang disederhanakan. AVL Tree dapat direpresentasikan dengan menggunakan array maupun linked list.

Pengurutan node secara manual.



**DAFTAR PERTANYAAN**

1. Operasi apa saja yang dapat dilakukan pada AVL Tree?
2. Berapakah kompleksitas ketika melakukan operasi insert dan delete pada AVL Tree?

**JAWABAN**

**JAWAB DI SINI!!!**

**LAB SETUP**

Hal yang harus disiapkan dan dilakukan oleh praktikan untuk menjalankan praktikum modul ini, antara lain:

1. Menyiapkan IDE untuk membangun program python (Spyder, Sublime, VSCode, dll);
2. Python sudah terinstal dan dapat berjalan dengan baik di laptop masing-masing;
3. Menyimpan semua dokumentasi hasil praktikum pada laporan yang sudah disediakan.

**ELEMEN KOMPETENSI I**

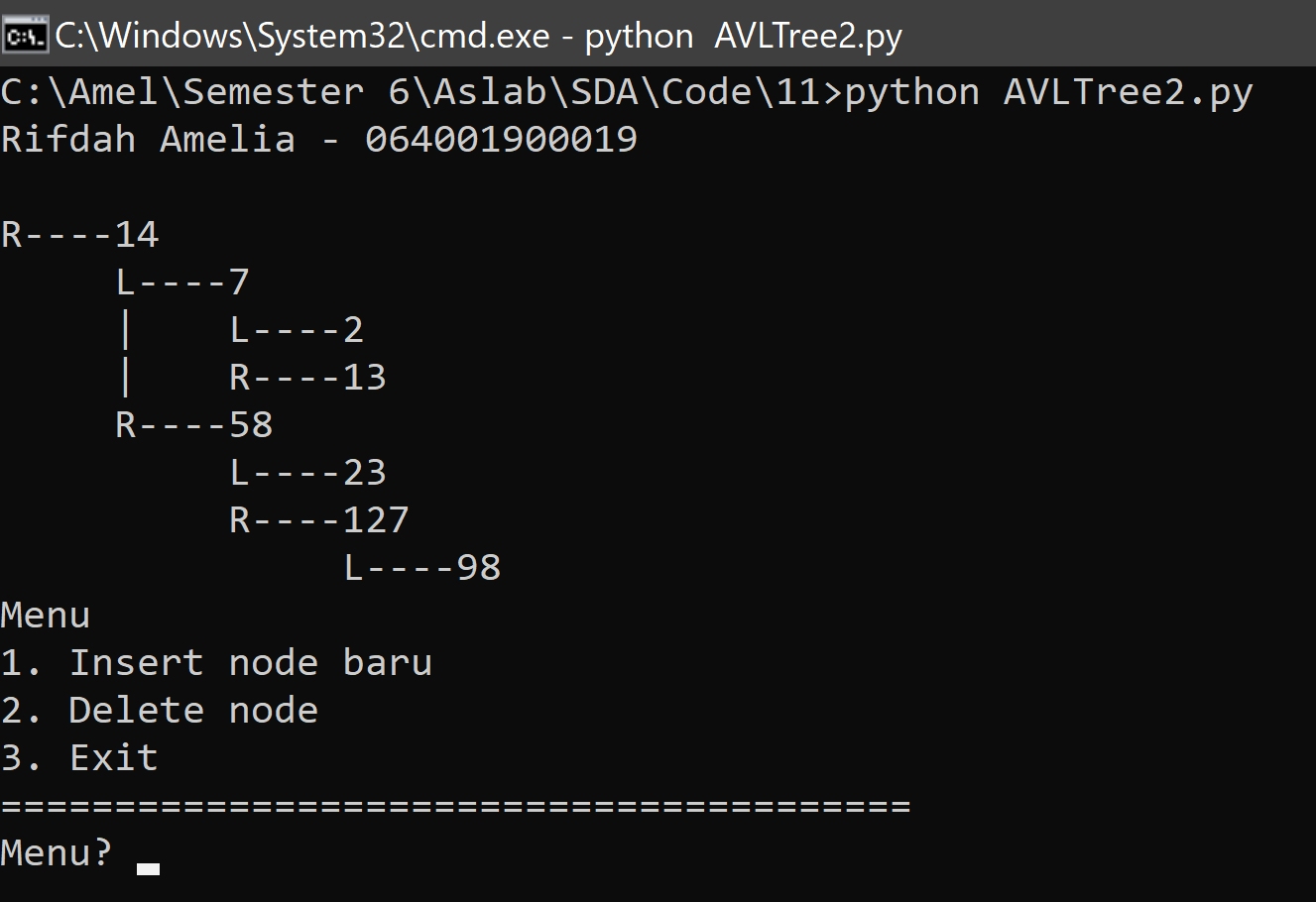
**Deskripsi :** Mampu membuat program tentang AVL tree sesuai perintah yang ada.

**Kompetensi Dasar :** Membuat program yang mengimplementasikan AVL tree

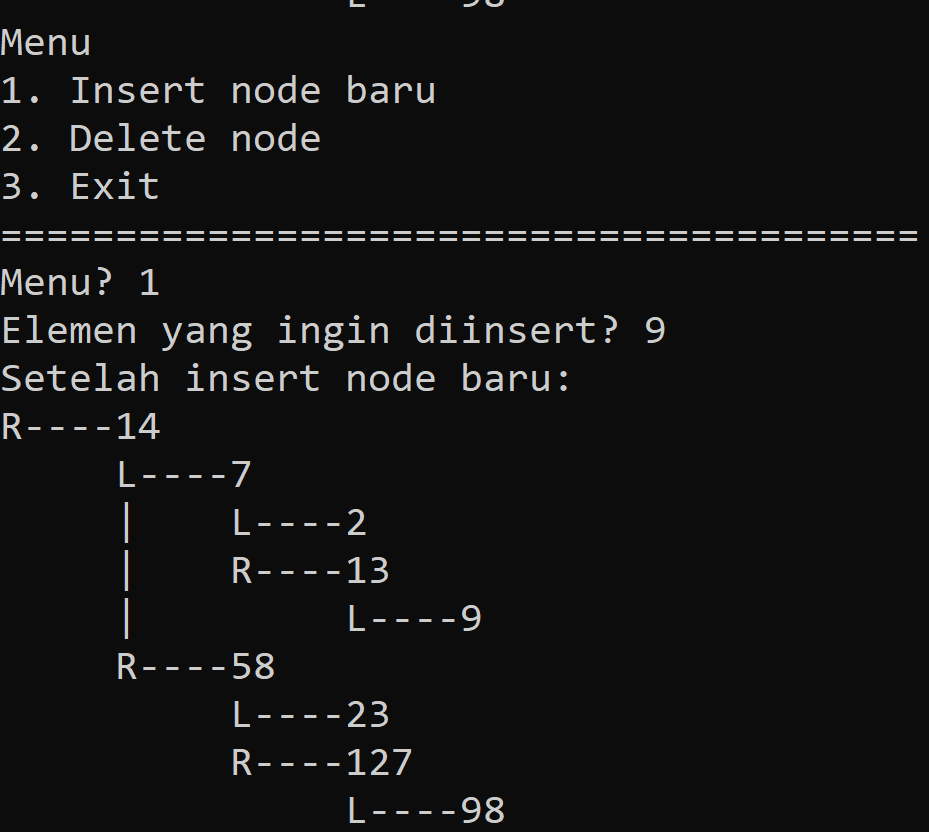
**LATIHAN 1**

* + - 1. Buatlah sebuah program yang mengimplementasikan AVL tree yang melakukan operasi insert dan delete node pada tree.
      2. Setiap program wajib menampilkan nama dan nim di bagian atas program.

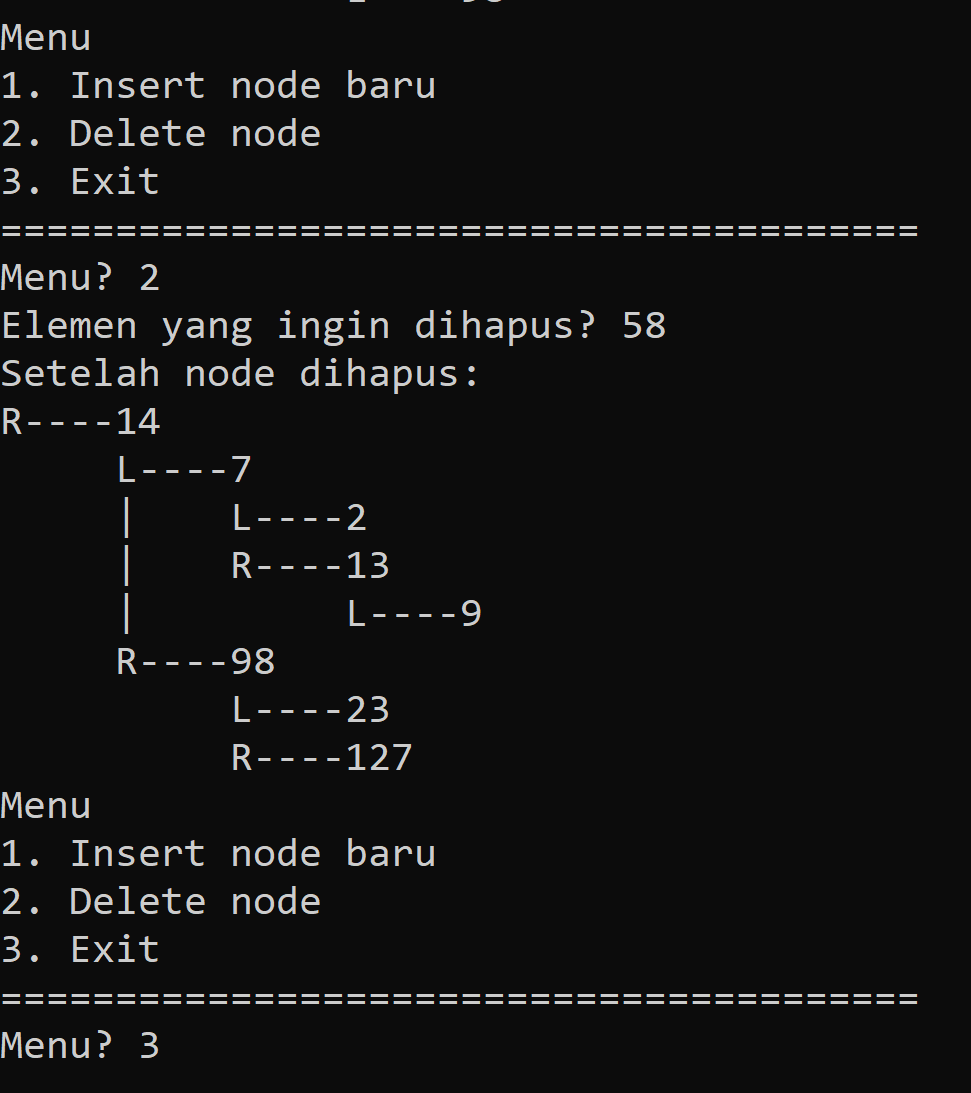
Kondisi ketika code baru dijalankan



Kondisi ketika memilih menu 1



Kondisi ketika memilih menu 2



Source Code

| import sys  class TreeNode(object):  def \_\_init\_\_(self, key):  self.key = key  self.left = None  self.right = None  self.height = 1  class AVLTree(object):  def insert\_node(self, root, key):  if not root:  return TreeNode(key)  elif key < root.key:  root.left = self.insert\_node(root.left, key)  else:  root.right = self.insert\_node(root.right, key)  root.height = 1 + max(self.getHeight(root.left),  self.getHeight(root.right))  balanceFactor = self.getBalance(root)  if balanceFactor > 1:  if key < root.left.key:  return self.rightRotate(root)  else:  root.left = self.leftRotate(root.left)  return self.rightRotate(root)  if balanceFactor < -1:  if key > root.right.key:  return self.leftRotate(root)  else:  root.right = self.rightRotate(root.right)  return self.leftRotate(root)  return root  def delete\_node(self, root, key):  if not root:  return root  elif key < root.key:  root.left = self.delete\_node(root.left, key)  elif key > root.key:  root.right = self.delete\_node(root.right, key)  else:  if root.left is None:  temp = root.right  root = None  return temp  elif root.right is None:  temp = root.left  root = None  return temp  temp = self.getMinValueNode(root.right)  root.key = temp.key  root.right = self.delete\_node(root.right,  temp.key)  if root is None:  return root  root.height = 1 + max(self.getHeight(root.left),  self.getHeight(root.right))  balanceFactor = self.getBalance(root)  if balanceFactor > 1:  if self.getBalance(root.left) >= 0:  return self.rightRotate(root)  else:  root.left = self.leftRotate(root.left)  return self.rightRotate(root)  if balanceFactor < -1:  if self.getBalance(root.right) <= 0:  return self.leftRotate(root)  else:  root.right = self.rightRotate(root.right)  return self.leftRotate(root)  return root    def leftRotate(self, z):  y = z.right  T2 = y.left  y.left = z  z.right = T2  z.height = 1 + max(self.getHeight(z.left),  self.getHeight(z.right))  y.height = 1 + max(self.getHeight(y.left),  self.getHeight(y.right))  return y  def rightRotate(self, z):  y = z.left  T3 = y.right  y.right = z  z.left = T3  z.height = 1 + max(self.getHeight(z.left),  self.getHeight(z.right))  y.height = 1 + max(self.getHeight(y.left),  self.getHeight(y.right))  return y  def getHeight(self, root):  if not root:  return 0  return root.height    def getBalance(self, root):  if not root:  return 0  return self.getHeight(root.left) - self.getHeight(root.right)  def getMinValueNode(self, root):  if root is None or root.left is None:  return root  return self.getMinValueNode(root.left)  def preOrder(self, root):  if not root:  return  print("{0} ".format(root.key),end=(','))  self.preOrder(root.left)  self.preOrder(root.right)  def printHelper(self, currPtr, indent, last):  if currPtr != None:  sys.stdout.write(indent)  if last:  sys.stdout.write("R----")  indent += " "  else:  sys.stdout.write("L----")  indent += "| "  print(currPtr.key)  self.printHelper(currPtr.left, indent, False)  self.printHelper(currPtr.right, indent, True)  myTree = AVLTree()  root = None  nums = [64,35,29,15,6,43]  for num in nums:  root = myTree.insert\_node(root, num)  myTree.printHelper(root, " ", True)  while True:  print("MENU :")  print("1.MASUKKAN NODE")  print("2.HAPUS NODE")  print("3.KELUAR")    ch = int(input("PILIHLAH: "))  if ch==1:  print('\n')  item = int(input("insert Node: "))  root = myTree.insert\_node(root, item)  myTree.printHelper(root, " ", True)    print('\n')  elif ch==2:  print('\n')  item = int(input("hapus Node: "))  root = myTree.delete\_node(root,item)  myTree.printHelper(root, " ", True)  elif ch==3:  print('Terimakasih')  sys.exit()  print("Maulana ikhlasul ahnaf // 064002100016")  myTree.printHelper(root, " ", True) |
| --- |

Screenshot

|  |
| --- |

**KESIMPULAN**

**Buatlah kesimpulan hasil praktikum modul ini!!! (MINIMAL 3 BARIS)**

**CEKLIST**

| 1. Memahami dan mengimplementasikan AVL tree pada Python | () |
| --- | --- |
|  |  |

**REFERENSI**

<https://www.geeksforgeeks.org/avl-tree-set-1-insertion/>

<https://algorithmtutor.com/Data-Structures/Tree/AVL-Trees/>