Öğrencinin

ADI: Ayşe Verda

SOYADI: Gülcemal

NUMARASI:g221210038

ŞUBESİ : 2/A

2.ÖDEV

Programımız veri.txt dosyasından okuduğu her satır sayı için bir AVL ağacı oluşturur. Oluşturduğu her AVL ağacının yaprak düğümü harici düğümlerini toplayarak toplamı Ascii karakterlerine aktararak ekranda Ascii karşılığı basar. Daha sonrasında her AVL ağacı için bir stack oluşturur. İlgili stacklara AVL ağaçlarının yaprak düğümlerini postorder okuma yolu ile yerleştirir.

Ödevi maalesef buraya kadar yapabildim. Devamında bir sayaç oluşturup tek sayılı steplerde tüm stack'ın top elemanlarının karşılaştırılıp en küçük olanının silinmesi ve sayacın artırılması; çift sayılı steplerde ise tüm stack'ın top elemanlarının karşılaştırılıp en büyük olanının silinmesi sayacın bir artırılması gerekiyor fakat bu kısmı koda dökemedim.

Program 2 ana dosyadan oluşuyor : main.cpp,avltree.cpp.

Avltree kodları ile AVL ağaçlarını oluşturdum, Ağaçların denge faktörlerini kontrol eden fonksiyonlar ekledim. AVL ağaçlarının ASCII kodlarını da buradaki fonksiyonlarla hesapladım. Yaprak düğümleri ile yığın işlemleri de burada bulunmakta.

Main kodları ile dosyadan verileri çektim ve fonksiyonları kullanarak çıktılarımı ürettim.

Silme işlemlerini tamamlayamasam da yigin kodlarını silmedim. Eksikler fakat ekleme , çıkarma gibi yığın fonksiyonlarımı burada bulabilirsiniz.

Ascii kodlarını yazdıran ve yığın işlemlerini yapan kodlar:

```
int AVLTree::postorderSumOfNonLeafNodes(AVLNode* node) {
    if (node == nullptr)
        return 0;

    int leftSum = postorderSumOfNonLeafNodes(node->left);
    int rightSum = postorderSumOfNonLeafNodes(node->right);

    if (node->left || node->right) {
        return leftSum + rightSum + node->data; // Include non-leaf nodes
    }

    return leftSum + rightSum;
}
void AVLTree::postorderLeafNodesToStack(AVLNode* node) {
    if (node == nullptr)
        return;

    postorderLeafNodesToStack(node->left);
    postorderLeafNodesToStack(node->right);

    if (node->left == nullptr && node->right == nullptr) {
        leafStack.ekle(node->data);
    }
}

void AVLTree::processLeafStack() {
    cout <<"bu avl agacinin yaprakları postorder sekilde yigina yerlesti"<<endl;
    while (leafStack.tepe != nullptr) {
        int leafData = leafStack.getir();
        // Process the leaf data as needed
        // For example, print or perform some operation
        cout << "  "<< leafData <<endl;

        // Remove the leaf node from the stack
        leafStack.cikar();
    }
```

Avl ağacının dengesini sağlayan kodlar:

```
28  AVLNode* AVLTree::rightRotate(AVLNode* y) {
29      AVLNode* x = y->left;
30      AVLNode* T2 = x->right;
31
32      x->right = y;
33      y->left = T2;
34
35      y->height = std::max(height(y->left), height(y->right)) + 1;
36      x->height = std::max(height(x->left), height(x->right)) + 1;
37
38      return x;
39  }
40
41  AVLNode* AVLTree::leftRotate(AVLNode* x) {
42      AVLNode* y = x->right;
43      AVLNode* T2 = y->left;
44
45      y->left = x;
46      x->right = T2;
47
48      x->height = std::max(height(x->left), height(x->right)) + 1;
49      y->height = std::max(height(y->left), height(y->right)) + 1;
50
51      return y;
52  }
53
```