Identificação: Pedro Mesquita Maia – 2312664

Descrição: O objetivo do trabalho foi construir uma árvore binária de busca (BST), utilizando uma lista de números inteiros e funções para inserir e percorrer os nós da árvore. Além disso, foi solicitado verificar se a árvore gerada é uma BST válida, e também para contar a quantidade de filhos de cada nó.

Estrutura: Foram criadas várias funções, além da main, para realizar as operações necessárias: createNode, insertNode, binaryTree, isBinarySearchTree, print_in_order_traversal, count_print_in_order_traversal, count, printIsBinarySearchTree, createValidBST, createEmptyTree, createSingleNodeTree, e runTests. Após a construção da árvore, foi chamada a função binaryTree para inserir os números na árvore e em seguida, a função print_in_order_traversal para percorrer e imprimir os nós da árvore.
Solução: Foram declaradas funções para criar nós, inserir elementos na árvore, verificar se ela é uma BST, e realizar um percurso in-order para imprimir e contar os filhos de cada nó. A função createNode é responsável por alocar um novo nó com o dado fornecido, enquanto insertNode insere um nó na árvore respeitando as regras de BST. A função binaryTree insere uma lista de números de forma ordenada na árvore.

A função print_in_order_traversal realiza um percurso in-order, percorrendo a árvore e imprimindo cada nó, seguido da contagem de seus filhos, que é realizada pela função count_print_in_order_traversal. Esta última função utiliza a função auxiliar count, que percorre a subárvore e retorna o número de filhos de um nó específico.

Após a criação e a impressão da árvore, a função isBinarySearchTree é utilizada para verificar se a árvore segue as regras de uma BST. Ela faz isso ao comparar o valor de cada nó com os limites mínimo e máximo permitidos e recursivamente chama a função para verificar as subárvores esquerda e direita. A função printIsBinarySearchTree imprime o resultado dessa verificação.

Observações: Eu não tinha me atentado na hora de fazer a verificação da BST e não estava pegando o maior e o menor valor da árvore para usar como validador de uma BST. Tive que procurar na internet pois não tinha linkado as ideias. Mas algo que me ajudou bastante a entenderfoi a implementação dos testes pois me fez pensar nos diferentes cenários, como uma árvore vazia, uma árvore com apenas um nó, e uma árvore completa válida. O código foi testado e funcionou certinho.

Para compilar foi utilizado o comando gcc -Wall -o main main.c.