**Question 1: Split a Binary Search Tree (BST)**You are given a Binary Search Tree (BST) and an integer value, V. Your task is to split the BST into two separate trees:

* **Tree 1:** This tree should contain all nodes with values less than or equal to V.
* **Tree 2:** This tree should contain all nodes with values greater than V.

Implement a function that splits the BST based on the given value, V. After the split, return the inorder traversal of both resulting trees.

**Question 2: AVL Tree with Insertions, Deletions, and Balancing**You are tasked with implementing an AVL tree and performing a series of operations on it. Specifically, you need to:

* **Insert Integers:** Insert the integers from 1 to 5 into an AVL tree, in the given sequence.
* **Delete Even Numbers:** After the insertion, delete all even integers (2 and 4) from the AVL tree, while ensuring that the tree remains balanced after each deletion.
* **Print Tree Details:** After each deletion, print:
  + The height of the AVL tree.
  + The inorder traversal of the AVL tree.

The function should handle both insertion and deletion of elements in the AVL tree. After each deletion, print the height and the inorder traversal of the tree.

**Question 3: Circular Priority Queue Using Arrays**You are tasked with implementing a circular priority queue using an array. This queue will store elements, where each element has a priority, and elements with higher priorities should be dequeued first. Your implementation should support the following operations:

* **Insertions (Enqueue):** Insert elements into the queue such that they are sorted by priority. When a new element is added, it should be placed in the correct position in the queue according to its priority.
* **Removals (Dequeue):** Remove the highest-priority element from the queue. The element with the highest priority should be dequeued first.
* **Circular Array:** Use a circular array with a fixed size of 10. Once the array reaches its capacity and more elements are added, it should wrap around to reuse the space.
* **Queue Order:** Maintain the order of the queue elements and ensure that the array is used efficiently, with no empty gaps between elements when possible.

**Question 4: Flatten a Binary Tree to a Linked List**Write a function that flattens a binary tree into a linked list in place, using the same tree nodes. The flattened tree should follow the preorder traversal. Preorder traversal is a type of depth-first traversal where you visit the node first, then the left child, and finally the right child. After flattening the tree:

* The left child of each node should be NULL.
* The right child should point to the next node in the preorder traversal.

Print the flattened tree in preorder as the resulting linked list.