Exercise 3: binary tree

TA: Ren, Peng

Date: 11/15/2023

Question 1:

1. **What is the purpose of this code?**

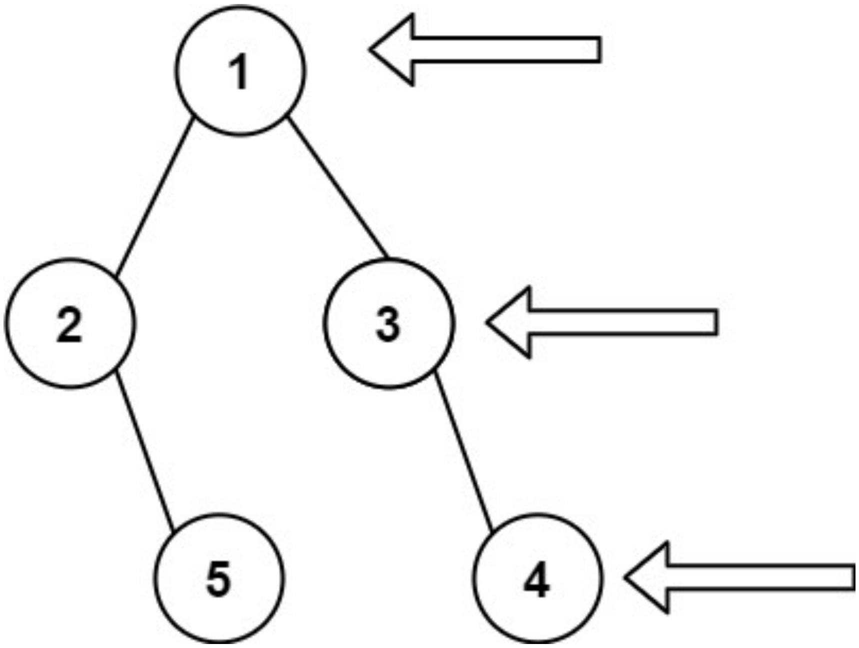
```c
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary tree node
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

void process_fuction(struct Node* root) {
    if (root == NULL)
        return;

    struct Node* queue1[1000];  // Assuming a maximum of 1000 nodes in the tree
    int front = 0, rear = 0;

    queue1[rear++] = root;

    while (front < rear) {
        struct Node* queue2[1000];
        int tempRear = 0;

        while (front < rear) {
            struct Node* cur_node = queue1[front++];

            printf("%d ", cur_node->data);

            queue2[tempRear++] = cur_node->left;

            queue2[tempRear++] = cur_node->right;
        }
        for (int i = 0; i < tempRear; i++) {
            queue1[i] = queue2[i];
        }

        rear = tempRear;
        front = 0;
    }
}
```

2. **The code above contains a bug. Please find it and fix it.**

   Hint: The same type of bug occurred twice in the code.

3. **What is the time complexity of the above code? Since there are two while loops, is the time complexity O(N^2)?**

4. **Given the `root` of a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.**



You don't need to write code. Just write down how you would get the results, which kind of algorithm you would use.

Question 2:

1. **Please finish the following code: insert a new node to the binary search tree:**

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a binary tree node
```

```
typedef struct Node {
    int data;
    struct Node *left;
    struct Node *right;
} Node;

Node* createNewNode(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

Node* insertIntoBST(Node* root, int value) {
    // add your code here!
}
```

2. **Time complexity of inserting a node into a binary search tree? Assume the tree height is h.**

Question 3:

1. **Given a sorted array, your task is to complete the following formula that can be used to build the binary search tree:**

```
Node* sortedArrayToBST(int arr[], int start, int end) {
    if (start > end) return NULL;

    int mid = start + (end - start) / 2;
    Node* root = createNewNode(arr[mid]);

    // Add your code here


    return root;
}
```