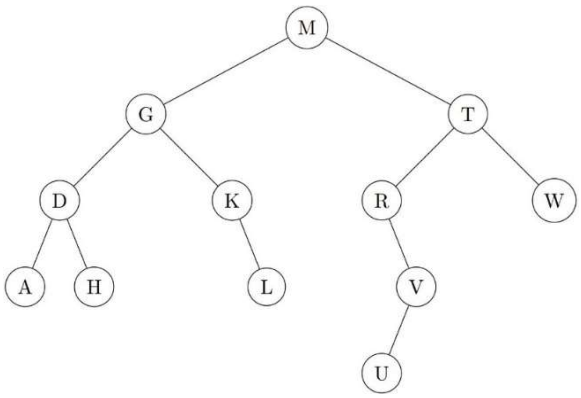
Assignment 2: Binary Search Tree and AVL Tree – Solution

Due: 3:00 pm, Mon., 11/21/2022

Binary Search Tree

Question 1 (30 pts.). Use the following binary search tree to answer the questions below.

(1.a) What is the pre-order traversal of this tree?
(1.b) What is the in-order traversal of this tree?
(1.c) What is the post-order traversal of this tree?

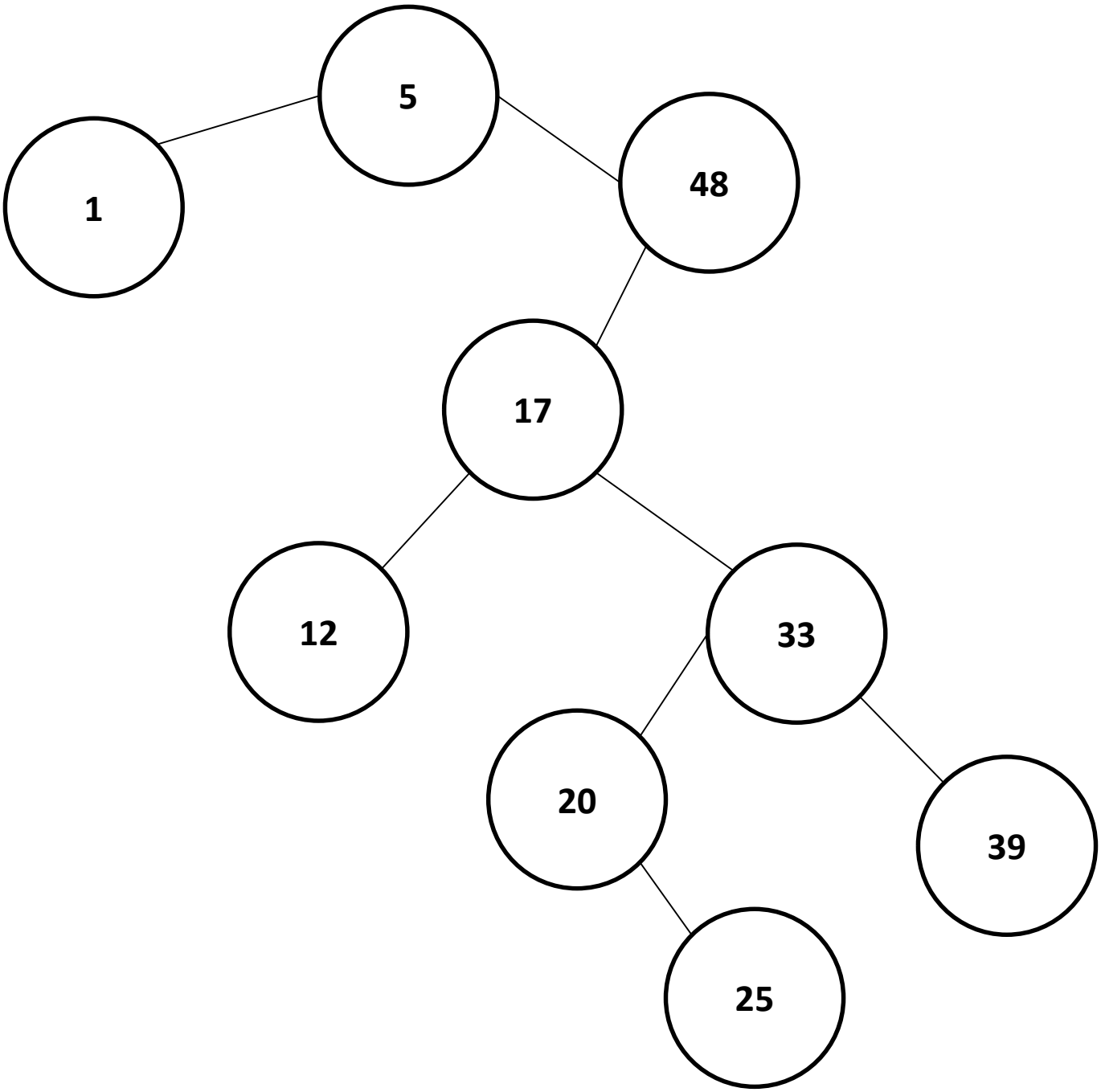


(1.a) Answer. M, G, D, A, H, K, L, T, R, V, U, W
(1.b) Answer. A, D, H, G, K, L, M, R, U, V, T, W
(1.c) Answer. A, H, D, L, K, G, U, V, R, W, T, M

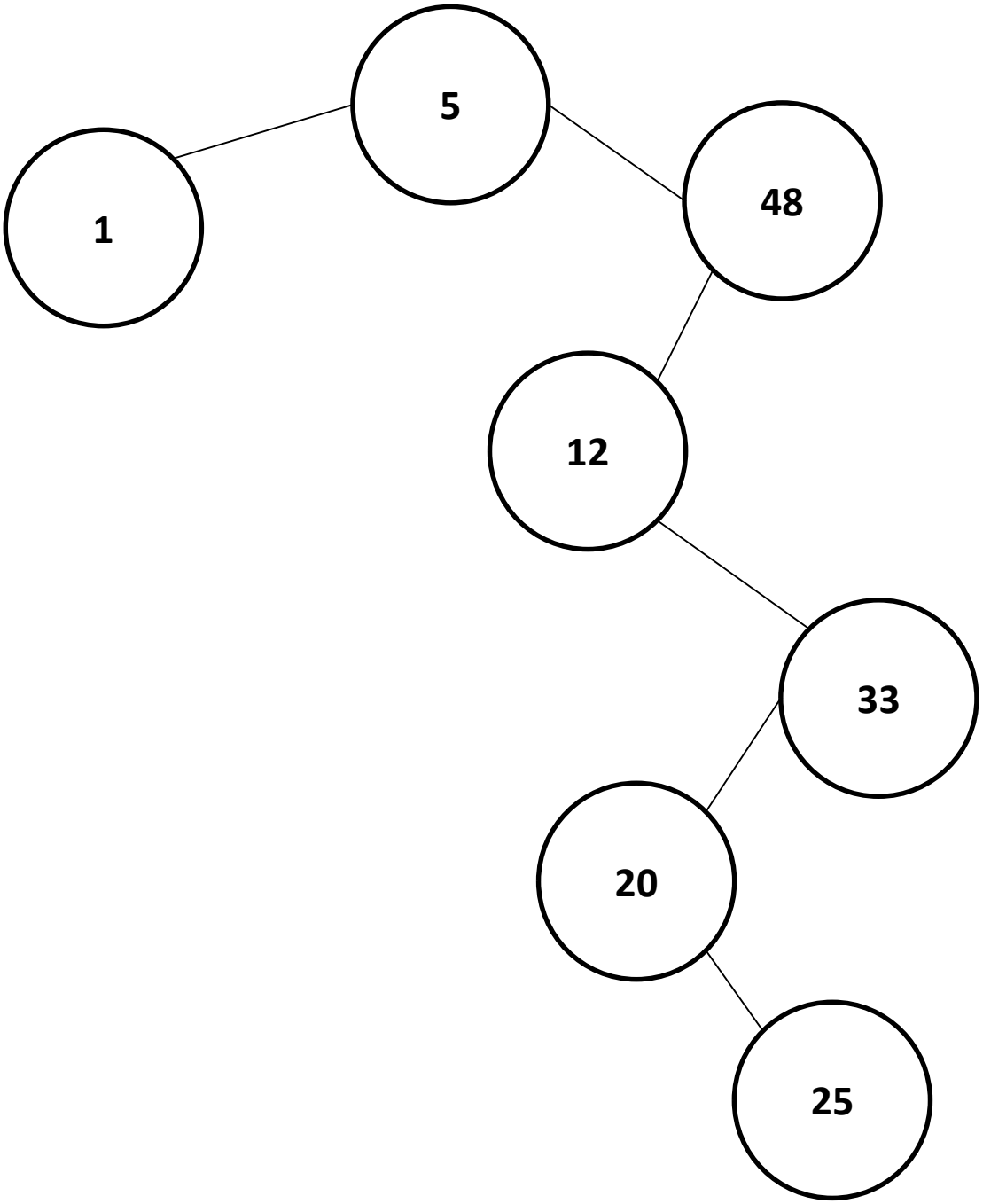
Question 2 (20 pts.). Execute the following computations and show them in figure format:

(2.a) Create a binary search tree and insert the following items into it:
{5, 48, 17, 1, 33, 20, 25, 12, 39}.
(2.b) Delete the following items from it: {17, 39}.

(2.a) Answer.

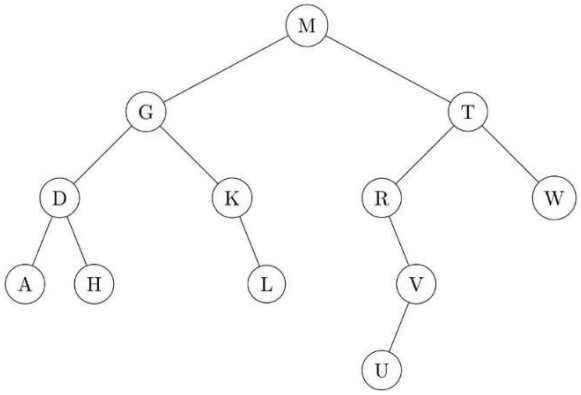


(2.b) Answer.



Question 3 (50 pts.). Write a program in C++ programming language to:

(3.a) Find the Smallest Node in a BST.
(3.b) Determine the Height of a BST.
(3.c) Determine the Number of Nodes.
(3.d) Create the Mirror Image of a BST.
(3.e) Delete a BST.



(3.a) Answer.

```
int smallestNode(struct node* node)
{
      struct node* current = node;
      while (current->left != NULL)
      {
            current = current->left;
      }
      return (current->key);
}
```

(3.b) Answer.

```
int nodeHeight(node* root)
{
      if (root == NULL)
            return 0;
      else
      {
            int leftHeight = nodeHeight(root->left);
            int rightHeight = nodeHeight(root->right);

            if (leftHeight > rightHeight)
                  return (leftHeight + 1);
            else
                  return (rightHeight + 1);
      }
}
```

(3.c) Answer.

```
int totalNodes(node* root)
{
	if (root == NULL)
		return 0;
	int left = totalNodes(root->left);
	int right = totalNodes(root->right);
	return left + right + 1;
}
```

(3.d) Answer.

```
void mirrorImage(struct node* node)
{
	if (node == NULL)
		return;
	else {
		struct node* root;
		mirrorImage(node->left);
		mirrorImage(node->right);
		root = node->left;
		node->left = node->right;
		node->right = root;
	}
}
```

(3.e) Answer.

```
void deleteTree(node* node)
{
	if (node == NULL) return;
	deleteTree(node->left);
	deleteTree(node->right);
	delete node;
}
```