Our Solution(s) Run Code

Solution 1 Solution 2

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <cmath>
#include <float.h>
using namespace std;

class BST {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val);
  BST &insert(int val);
};

int findClosestValueInBst(BST *tree, int target);
int findClosestValueInBstHelper(BST *tree, int target, double closest)

// Average: O(log(n)) time | O(1) space
// Worst: O(n) time | O(1) space
int findClosestValueInBst(BST *tree, int target) {
  return findClosestValueInBstHelper(tree, target, DBL_MAX);
}

int findClosestValueInBstHelper(BST *tree, int target, double closest)
  BST *currentNode = tree;
  while (currentNode != NULL) {
    if (abs(target - closest) > abs(target - currentNode->value)) {
      closest = currentNode->value;
    }
    if (target < currentNode->value) {
      currentNode = currentNode->left;
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
class BST {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val);
  BST &insert(int val);
};

int findClosestValueInBst(BST *tree, int target) {
  // Write your code here.
  return -1;
}

```

Our Tests

Custom Output Submit Code

```
  if (val > value) {
    if (left == NULL) {
      BST *newBST = new BST(val);
      left = newBST;
    } else {
      left->insert(val);
    }
  } else {
    if (right == NULL) {
      BST *newBST = new BST(val);
      right = newBST;
    } else {
      right->insert(val);
    }
  }
```

Run or submit code when you're ready.