**Data Structures**

**Spring 2020**

**Lab #4\_Problem 1**

**Date: 2020. 05. 19**

**ID/Name: 201533661 이승수**

**[Lab4\_Problem1(kd tree)] (20 points)**

**hand-trace the source code for inserting the first two of the 6 input points used in the source code.**

**(ignore the “find\_median” function except for the call and return from the function; compute the median by hand)**

**input point(wp): [**(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)]

**[Result]**

struct kd\_node\_t wp[] = {

{{2, 3}}, {{5, 4}}, {{9, 6}}, {{4, 7}}, {{8, 1}}, {{7, 2}}

}; // median= {{5,4}}, {{7,2}}

//gloabal variables

int visited;

#define N 1000000

#define rand1() (rand() / (double)RAND\_MAX)

#define rand\_pt(v) { v.x[0] = rand1(); v.x[1] = rand1(); v.x[2] = rand1(); }

root = make\_tree(wp, sizeof(wp) / sizeof(wp[1]), 0, 2);

//struct kd\_node\_t\* make\_tree(struct kd\_node\_t\* t, int len, int i, int dim)

if ((n = find\_median(wp, wp + sizeof(wp), 0))) { // median={2,3}

i = (i + 1) % dim; //i=1

n->left = make\_tree(t, n - t, i, dim); // {2,3}->left= 0

if (!len) return 0;//if new node is null, return 0

n->right = make\_tree(n + 1, t + len - (n + 1), i, dim); // {2,3}->right=0

if (!len) return 0;//if new node is null, return 0

}

n->right = make\_tree(n + 1, t + len - (n + 1), i, dim); //n->right= 0

}

return n; //n={2,3}-(left=null, right=null)

visited = 0;

found = 0;

nearest(root, &testNode, 0, 2, &found, &best\_dist);

//…continue to next question.

**Call of the nearest function (including the input parameters and output parameters)**

**\*\*pretend new node is (2,3)**

**[Result]**

nearest(root, &testNode, 0, 2, &found, &best\_dist);

//root=(5,4), testNode=(9,2), found=0, best\_dist=0

double d, dx, dx2;

d = dist({{5,4}},{{9,2}}, 0, 2); //d=(9-5)^2 + (2-4)^2 = 20

dx = root->x[i] - nd->x[i]; // dx= 5-2 = 3

dx2 = dx \* dx; // dx2= 3\*3 = 9

visited++; //visited=1

if (!\*best || d < \*best\_dist) { //best=0(found), d(20) > best\_dist(0)

\*best\_dist = d; //best\_dist=20

\*best = root; //best=(5,4)

}

if (!\*best\_dist) return; //best\_dist=20, so nearest function end