Computer Science 1 – Program 5
KnightsHoc Internet Share (Binary Search Trees)
Assigned: 3/21/12
Due: 4/4/12 (Wednesday) by 11:55pm (WebCourses time)

Purpose
Learn to implement the functionality of a binary search tree and to combine that functionality with linked lists. Of course, more and more practice with pointers!!!

The Problem
While the UCF, university-wide wireless internet connection often works fine, many students complain that the range of said connection is shoddy, at best, and a group of "techy" CS students have come up with a solution: *KnightsHoc Internet Share* (*KIS* for short). All students participating in *KnightsHoc Internet Share* must allow their laptops to act as Ad-Hoc Wireless Networks (aka Virtual Routers or Wireless Hotspots), thereby extending the "range" of the UCF wireless internet connection. Note: although all participating students must allow their laptops to act as Wireless Hotspots for other students, each student (owner of laptop and administrator of their own laptop's Wireless Hotspot) can choose whose laptops can (or cannot) connect (LINK) to their Wireless Hotspot.

> Example:
> Students A, B, and C all participate in the *KnightsHoc Internet Share* program. Let us assume that there is an official, UCF hotspot in HEC with a range of 200 feet. Students B and C, whose laptops are 240 feet from said hotspot in HEC, will have no internet access, as they are forty feet out of range. However, if Student A is sitting 190 feet away from the UCF hotspot in HEC and, therefore, has internet connection, Students B and C could possibly still get wireless access **IF** Student A has allowed them access to his/her Wireless Hotspot **and IF** they are in range of said Hotspot.

If you have a wireless card, under the "Set Up a Connection or Network" in Windows 7, you will see a new option: "Set up a wireless ad hoc (computer-to-computer) network". At first glance, this option would seem to allow for the proposed environment; unfortunately, it only allows one to wirelessly share a wired internet connection (or you need two wireless cards, which must of us don't have). As such, the aforementioned "techy" CS students put their heads together and wrote a lightweight program, *KnightsHoc Internet Share*, which allows a laptop to wirelessly share a **wireless** connection. Point being, we assume the technology exists for the sake of this assignment.

As shown in the example above, there are two conditions for Student B to share the internet connection from Student A's Wireless Hotspot:

1) They must be LINKed together, meaning that Student B must be added to the allowed list of those who can access Student A's Wireless Hotspot

2) Student B must be within the broadcast range of Student A's Wireless Hotspot.

   Note: range is solely determined based off of the broadcasting capability of the laptop broadcasting the Wireless Hotspot. In this example, if Student A's broadcast range was 25 feet, only those students within 25 feet of Student A could connect to Student A's Wireless Hotspot.

For the sake of this assignment, each laptop will have a limited **broadcast** range. **However**, we will assume that all laptops have an unlimited range with respect to accessing the Wireless Hotspot of another student. Basically, this stipulation simplifies the problem, limiting the range calculation to only the range of the broadcasting laptop.

To be clear, we give one final example:
Student A and Student B both participate in the *KnightsHoc Internet Share* program, and they are LINKed together. Assume they are 20 feet apart, and the broadcast ranges of Student A and B are 30 feet and 15 feet, respectively. Therefore, Student B could use the internet connection of Student A, because Student B is within the 30 ft broadcasting range of Student A. However, Student A could not use the internet connection of Student B, because Student A is not within the 15 ft broadcasting range of Student B.

Range Calculation:
You will determine the range between two laptops based on the Euclidean distance between them. As a reminder, the 2D Euclidean distance between objects, p and q, is given by:

$$EucDis(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Your assignment is to simulate these laptops as they move around a 2D grid (x/y coordinate plane) and connect to each other.

Implementation
All laptops will be stored in a Binary Search Tree based off of their unique MAC address (just an `int`). Each laptop (node) will have several data members, including the MAC address, name of student, x and y coordinates, and a linked-list that will maintain a list of all laptops that it is LINKed to.

You will need to make use of the following structures:

```
typedef struct allowedMACs {
     int linkedMAC;
     struct allowedMACs *next;
} links;
```

This struct is a LINKED-LIST. Every laptop must maintain a linked list of the laptops that are LINKed to it (allowed to connect). you will use this struct as a member of the BSTnode struct.

```
typedef struct tree_node {
     int MAC;
     char firstName[21];
     char lastName[21];
     int broadcastRange;
     int x;
     int y;
     links *deviceLinks;
     int numLinks;
     struct tree_node *left;
     struct tree_node *right;
} BSTnode;
```

This struct will be used to store each laptop's information. All members of this struct should be self-explanatory, with the possible exception of:
`links *deviceLinks;`

That member, as mentioned above, represents the head of a linked list of this particular laptop's links (allowed connections).

Your *KnightsHoc Internet Share* program will read in a series of instructions from a file, such as JOIN, QUIT, LINK, UNLINK, etc., and you will need to perform the instruction and print the appropriate information to the output file.

<u>Input/Output Specifications</u>
The input file will be called "`KISsimulator.in`". The first line of the file will be a single integer that will be used to <u>seed your random numbers</u> (see `MOVEDEVICES` command for more info). The second line of the file will have two integers, sizeX and sizeY, representing the size of our 2D coordinate plane. The third line of the file will be an integer, *k*, indicating the number of commands that will be run on the database. The following *k* lines will each be a single command, **<u>with the relevant data on the same line</u>**, separated by spaces.

The commands you will have to implement are as follows:

- **`JOIN –`** A new student/laptop is joining the KIS program (a new node added). The command will be followed by the following information: MAC, a non-negative int, which you will use to sort the devices by; first and last name of student, each no longer than 20 characters; broadcastRange, a non-negative int, which represents the maximum broadcast range of the device; and X and Y, the initial X and Y coordinates of the device. If this student is not already participating in the *KnightsHoc Internet Share* program, they should be added to the BST. Otherwise, an error should be printed (see output).
- **`FINDMAC –`** Find a device based on its MAC address. This command will be followed by a MAC address to search for. If found in the BST, this command should print the MAC address, the student's name, the current X and Y coordinates of the device, and the number of links it currently has. If said device cannot be found, an error message should be printed (see output).
- **`LINK –`** This will be followed by two MAC addresses (two `ints`). When you see this command, this means that the two devices are now linked, and thus you should add the MAC address of EACH device to the other's linked list of linked devices (`deviceLinks`). All connections in the database are always two-way – e.g. if Elliott's laptop MAC is in Tony's linked-list of linked (allowed) devices, then Tony's laptop MAC is also in Elliott's linked-list of allowed devices. If either MAC in the `LINK` command is not in the database, or if they are already linked, an error message should be displayed (see output).
- **`UNLINK –`** This is the opposite of `LINK`. Like `LINK`, it will be followed by two MACs. You will have to find both devices in the BST, and remove their MACs from each others' linked-list. Again, if either device is not in the database, or they are not actually linked, an error message should be displayed (see output).
- **`QUIT –`** This command is followed by a single MAC address, and it means the user is quitting the *KnightsHoc Internet Share* program (so you must delete this laptop from the BST). NOTE: Before you do so, however, you must remove said MAC address from any other laptop's linked-list of allowed laptops. Thus, for the MAC address you are deleting, you will need to loop through its own linked-list of allowed laptops, and run the `UNLINK` command for each laptop in the list, thereby removing them from each others' lists. If the device you are trying to delete is not in the database already, an error message should display (see output).
- **`SHOWCONNECTIONS –`** This command is followed by a single MAC address. The command prints the MAC address, the student's name, first and then last, the number of links they have, the number of currently **<u>active</u>** connections (linked and in range), and then it prints a numerically ascending list of the MAC address that are currently active, along with other pertinent data of those MAC addresses. If said MAC address is not found in the database, an error message should be printed (see output). NOTE: Student A's active connections are defined as those MAC addresses that are (1) found in the linked-list of allowed laptops and (2) that are within the broadcasting range of Student A's laptop.

- **`PRINTKISMEMBERS` –** This command prints all participants, in the *KnightsHoc Internet Share* program, in ascending numerical order (based on MAC addresses), along with printing basic information about these devices (see output).
- **`MOVEDEVICES` –** This command pseudo-randomly assigns new values to the X and Y coordinates of each device, thereby simulating the random movement of the students/laptops as they travel around campus, resulting in them possibly entering and exiting various Wireless Hotspots. Translating to code, this command has you travel to each node of the BST, in ascending order based on MAC address. At each node, you first assign a randomly generated X coordinate and then you assign a randomly generated Y coordinate.

  As mentioned at the top of this section, the first line of the input file will be an integer that will be used to seed your random numbers; think of this as a starting point for the pseudo-random number generator. So if the seed integer was 12345, in code, you would "seed" the generator as follows:

  ```
  srand(12345);
  ```

  This simple line of code seeds the generator appropriately. Now every time you call `rand()`, it will return a pseudo-random number, which can be predetermined if the seed is known, and every time you run the program with that seed, the numbers generated will be exactly the same. Assuming `currentNode` points to a given node in the tree, the following two lines of code would be used to assign new X and Y coordinates, respectively:

  ```
  currentNode->x = rand() % sizeX;
  currentNode->y = rand() % sizeY;
  ```

  Where `sizeX` and `sizeY` are the sizes of the X and Y coordinate plane, respectively. As such, a student (device) can move to any location in the X/Y coordinate plane based off of a `MOVEDEVICES` command.

  Note: You need to include stdlib.h in your file in order to use rand() and srand().

Your program must output to a file, called "`KISsimulator.out`". **You must follow the program specifications exactly**. You will lose points for formatting errors and spelling.

*****WARNING*****
Your program MUST adhere to this EXACT format of the output shown below (spacing capitalization, use of dollar signs, periods, punctuation, etc). The graders will use very large input files, resulting in very large output files. As such, the graders will use text comparison programs to compare your output to the correct output. If, for example, you have two spaces between in the output when there should be only one space, this will show up as an error even through you may have the program correct. You WILL get points off if this is the case, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as the graders will be forced to go to text editing of your program in order to give you an accurate grade. So as an example, here is the first format shown above (if the instruction was an addition):

Again, your output MUST ADHERE EXACTLY to the sample output shown below.

Sample Input (read from a file called “**KISsimulator.in**”)

```
47567
50 50
47
PRINTKISMEMBERS
QUIT 56784
MOVEDEVICES
SHOWCONNECTIONS 56784
JOIN 56784 Homer Simpson 20 14 22
JOIN 65324 Marge Simpson 30 20 19
JOIN 35477 Bart Simpson 30 18 17
JOIN 22379 Ned Flanders 50 34 46
UNLINK 56784 65324
LINK 56784 65324
LINK 56784 35477
LINK 23493 35477
FINDMAC 35477
FINDMAC 56784
FINDMAC 65324
JOIN 12352 Maude Flanders 50 35 42
JOIN 56784 Homer Simpson 20 14 22
SHOWCONNECTIONS 35477
LINK 22379 12352
LINK 22379 56784
LINK 56784 65324
MOVEDEVICES
FINDMAC 56784
FINDMAC 22379
FINDMAC 12352
LINK 56784 12352
FINDMAC 56784
SHOWCONNECTIONS 56784
MOVEDEVICES
UNLINK 56784 22379
SHOWCONNECTIONS 56784
SHOWCONNECTIONS 65324
UNLINK 56784 65324
SHOWCONNECTIONS 65324
FINDMAC 56784
FINDMAC 22379
FINDMAC 77777
QUIT 35477
QUIT 12352
MOVEDEVICES
SHOWCONNECTIONS 22379
UNLINK 12352 22379
UNLINK 12352 35477
FINDMAC 22379
FINDMAC 12352
LINK 56784 65324
PRINTKISMEMBERS
```

Corresponding Output (saved to file called “**KISsimulator.out**”)

```
Cannot Perform PRINTKISMEMBERS Command:
      There are currently no participants of the KIS system.
Cannot Perform QUIT Command:
      MAC 56784 not found in the KIS system.
Cannot Perform MOVEDEVICES Command:
      There are currently no participants of the KIS system.
```

```
Cannot Perform SHOWCONNECTIONS Command:
      MAC 56784 - This MAC Address is not in the KIS system.
Homer Simpson, MAC 56784, joined KIS.
Marge Simpson, MAC 65324, joined KIS.
Bart Simpson, MAC 35477, joined KIS.
Ned Flanders, MAC 22379, joined KIS.
Cannot Perform UNLINK Command:
      MAC 56784 and MAC 65324 are not currently linked.
MAC 56784 and MAC 65324 are now linked.
MAC 56784 and MAC 35477 are now linked.
Cannot Perform LINK Command:
      MAC 23493 - This MAC Address is not in the KIS system.
Found:  MAC 35477, Bart Simpson, currently at position (18, 17), 1 Link
Found:  MAC 56784, Homer Simpson, currently at position (14, 22), 2 Links
Found:  MAC 65324, Marge Simpson, currently at position (20, 19), 1 Link
Maude Flanders, MAC 12352, joined KIS.
Cannot Perform JOIN Command:
      MAC 56784, Homer Simpson - already a participant in the KIS program.
Connections for MAC 35477, Bart Simpson, currently at position (18, 17):
      There are a total of 1 link(s).
      There are 1 active link(s) within the broadcast range of 30:
            MAC 56784, Homer Simpson, currently at position (14, 22)
MAC 22379 and MAC 12352 are now linked.
MAC 22379 and MAC 56784 are now linked.
Cannot Perform LINK Command:
      MAC 56784 and MAC 65324 are already linked.
All devices successfully moved.
Found:  MAC 56784, Homer Simpson, currently at position (48, 33), 3 Links
Found:  MAC 22379, Ned Flanders, currently at position (6, 23), 2 Links
Found:  MAC 12352, Maude Flanders, currently at position (0, 30), 1 Link
MAC 56784 and MAC 12352 are now linked.
Found:  MAC 56784, Homer Simpson, currently at position (48, 33), 4 Links
Connections for MAC 56784, Homer Simpson, currently at position (48, 33):
      There are a total of 4 link(s).
      There are 0 active link(s) within the broadcast range of 20:
All devices successfully moved.
MAC 56784 and MAC 22379 are no longer linked.
Connections for MAC 56784, Homer Simpson, currently at position (33, 8):
      There are a total of 3 link(s).
      There are 1 active link(s) within the broadcast range of 20:
            MAC 12352, Maude Flanders, currently at position (20, 15)
Connections for MAC 65324, Marge Simpson, currently at position (46, 34):
      There are a total of 1 link(s).
      There are 1 active link(s) within the broadcast range of 30:
            MAC 56784, Homer Simpson, currently at position (33, 8)
MAC 56784 and MAC 65324 are no longer linked.
MAC 65324 has no links.
Found:  MAC 56784, Homer Simpson, currently at position (33, 8), 2 Links
Found:  MAC 22379, Ned Flanders, currently at position (26, 31), 1 Link
MAC 77777 not found in the KIS system.
MAC 35477 has been removed from the KIS system.
MAC 12352 has been removed from the KIS system.
All devices successfully moved.
MAC 22379 has no links.
Cannot Perform UNLINK Command:
      MAC 12352 - This MAC Address is not in the KIS system.
Cannot Perform UNLINK Command:
      MAC 12352 - This MAC Address is not in the KIS system.
      MAC 35477 - This MAC Address is not in the KIS system.
```

```
Found:  MAC 22379, Ned Flanders, currently at position (28, 49), 0 Links
MAC 12352 not found in the KIS system.
MAC 56784 and MAC 65324 are now linked.
Members of KnightsHoc Internet Share:
      MAC 22379, Ned Flanders, currently at position (28, 49), 0 Links
      MAC 56784, Homer Simpson, currently at position (2, 6), 1 Link
      MAC 65324, Marge Simpson, currently at position (11, 0), 1 Link
```

Deliverables

Turn in a single file, *KISsimulator.c*, over WebCourses that solves the specified problem.

Grading Details

Your program will be graded upon the following criteria:

1) Adhering to the implementation specifications described here.
2) Your algorithmic design.
3) Correctness.
4) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (We're not kidding here. If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
5) Compatibility to CodeBlocks. (If your program does not compile in CodeBlocks, you will get a sizable deduction from your grade.)
7) **Use of Binary Search Trees with Linked Lists. Since the purpose of this assignment is to teach you to Binary Search Trees, you will not receive credit if you don't use them.**

Restrictions

Name the file you create and turn in *KISsimulator.c*. Although you may use other compilers, your program must compile and run using CodeBlocks. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments throughout your code, when appropriate.