**Assignment 4 Report, by Adam Berlak, ID: 30008230**

**User Manual:**

To compile and run the program write the following in the terminal:

1. gcc a4.c –o a4 –lm
2. ./a4

**How to use:**

The code in the main portion of my project is comprised of two sections. The first section interprets the user specified constants and generates a simulation based on them, presenting the number of idle, and successful probes, collisions and total probes. To customize the simulation modify the initializing variables. Changing N alters the size of the tree, and L corresponds to the number of levels. K specifies how many stations are ready and the program generates K random ready stations according to this number, depending on the seed, these values can both be changed. Lastly Start specifies which level the probing will start. The second portion of my code generates all possible outcomes by cycling through each possible K value for each level and prints out the result as a table in the terminal.

Note: For testing I recommend commenting out one section while focusing on another since testing the first section on many cases will require generating the second section over and over again.

**Function Descriptions:**

The generateTree functions creates a 2d array of strings, where each level is an array and contains the station numbers represented as strings. The probe function recursively checks each level of the tree based on the global constants.

**Observations:**

In my program from start = 0 to 8, the percentage of successful probes compared to total probes caps at 50 percent, while at start = 10, which is the last level, the program is able to achieve a 100 percent successful probe rate. Despite this, using a starting point of 10 is inefficient because for the majority of K values, the percentage of idle probes is well over 50 percent. In comparison, a start at level 0 is less likely to be idle but has a max 50 percent success rate with a K = 1024, and the collisions are much more frequent. In conclusions, as the starting level increases, the percentage of successful probes increases, the percentage of idle probes increases, and the percentage of collisions decreases. As the K value is increased, the percentage of successful probes increases, the percentage of idle probes decreases, and the percentage of collisions increases.

\*Numeric tables generated in Terminal through my program.\*

Testing done through Linux virtual machine