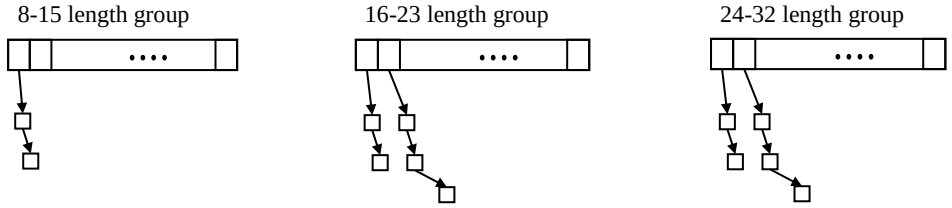
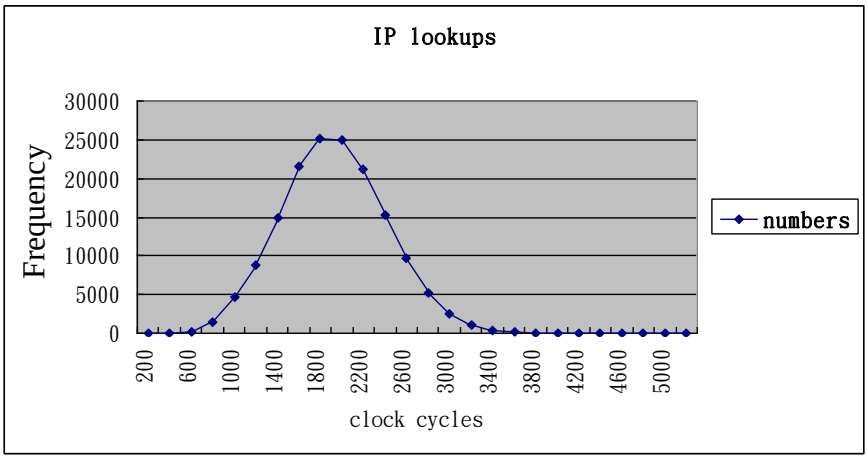
C Language Programming: Homework #8
Assigned on 12/17/2019(Tuesday), Due on 01/06/2020(Monday)

1. Write a program that can read data (called *prefix*) from a IP table to build a data structure (called *routing table*) stated below and then perform search, insert, and delete operations based on their corresponding traces. The trace of search contains IP addresses and the traces of insert and delete contain prefixes. Therefore, we will provide the sample prefix sets and three traces.
2. The format of IP addresses is a.b.c.d where a/b/c/d are 8-bit unsigned numbers. The format of prefixes (or called IP prefixes) is a.b.c.d/*m* where *m* is a number in the range of 8-32 and *m* can be stored in a 8-bit unsigned variable. Also, if the prefix is of length 8, 16, 24, or 32, it is represented in a short-hand format i.e., the length part may be omitted, like 4.0.0.0 instead of 4.0.0.0/8 and 4.17.255.0 instead of 4.17.255.0/24.
3. *Routing table:*
   The prefixes are divided into three groups of lengths 8-15, 16-23, and 24-32 and stored in 3 arrays called *segmentation tables* of sizes 2^8, 2^{12}, and 2^{12} as follows. For the group of prefixes of length 8-15, we use the first 8 significant bits of the prefixes as the array index to insert it into the segmentation table. For example, prefix 4.0.0.0/8 will be inserted in the linked list pointed to by the 4^{th} element of the segmentation table. For the group of prefixes of length 16-23 or 24-32, we use the first 12 significant bits of the prefixes as the array index to insert it into the segmentation table. For example, prefix 4.17.255.0/24 will be inserted in the linked list pointed to by the 65^{th} element of the segmentation table because the 12 MSB of 4.17 (00000100 00010001) is 65 (000001000001).
   IP Prefix Format：IP / Length (e.g.：4.0.0.0 4.17.255.0/24)



4. (50%)Use the trace of ip addresses to perform search (longest prefix match) and print out success and fail search times, repectively.
5. (20%)For insertion, use the trace of insert prefixes and for each prefix, and record the avg. insertion time. After insertion, use the trace of ip addresses to perform search (longest prefix match) and print out success and fail search times, repectively.
6. (20%)For deletion, use the trace of delete prefixes and for each prefix, record the avg. deletion time
   After deletion, use the trace of ip addresses to perform search (longest prefix match) and print out success and fail search times, and search times in terms of the number of *clock cycles.*
7. (10%)Compute the average search clock cycles, average insert clock cycles, average delete clock cycles, and use excel to draw the curve like the following figure, respectively.
   (Note: Therefore, there will be 3 figures in report)



8. updata your report to server, otherwise you will get -10 point.
9. If you will not submit your report , you get 0 point.

Command Line(You must use Parameter argc and argv)
./hw8 prefix_10K.txt trace_IPaddress_100K.txt insert_1K.txt delete_1K.txt
(Please Follow this Sequence，otherwise you will get -20 point)

Example
> ./hw8 prefix_10K.txt trace_IPaddress_100K.txt insert_1K.txt delete_1K.txt
After seg. table create
success search times= ?
fail search times= ?

After insertion
avg. insertion time= ? cycles
success search times= ?
fail search times= ?

After deletion
avg. deletion time= ? cycles
avg. search times=? cycles
success search times= ?
fail search times= ?

Note: If you want to draw the graph, you need to record the number of searches per 100 clocks, examples are as follows(Not really data):
100 : 0
200 : 257
300 : 46393
400 : 258326
500 : 85661
600 : 7570
700 : 4987
800 : 3055
900 : 593
1000 : 236
1100 : 79
1200 : 32
1300 : 18
1400 : 6
1500 : 1

1600 : 1
1700 : 1
1800 : 0
1900 : 0
2000 : 0

Note: How to compute clock cycle

RDTSC 教學 Example

```
unsigned long long int begin,end,total=0;
static __inline__ unsigned long long rdtsc(void)

{

unsigned hi, lo;

__asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));

return ( (unsigned long long)lo)|( ((unsigned long long)hi)<<32 );

}


int main(){
  begin=rdtsc();
  //欲測試函數
  end=rdtsc();
  total = end – begin; // total 即為 cpu clock cycle
}
```