Problem HT-4 (3 parts)  

**Part A** Consider a hash table that uses 21 buckets, each containing an unsorted LIFO list of items. Inserted entries are appended to the end of the bucket list. Suppose the hash table contains 210 entries total and the entries are evenly distributed across the hash table buckets. Assume that computing the hash function takes an average of two operations and comparing two strings takes an average of twenty operations. Ignore effects of spatial and temporal reference locality. Suppose that 80% of keys looked up are found in the hash table and 20% are not found. How many of these operations would be required for the average lookup in the hash table described above? (show work)

\[
k = \text{average bucket length} = \frac{210}{21} = 10 \text{ items/bucket}
\]

\[
T_a = 2 + 20 \left[ .8 \left( \frac{10+1}{2} \right) + .2 \times 10 \right] = 2 + 20[4.4 + 2] = 2 + 20[6.4] = 2 + 128 = 130 \text{ operations}
\]

130 operations

**Part B** Suppose the hash table automatically resizes (with the same 210 entries and found key probabilities) so that the average access time becomes 82 operations. How many hash table buckets are being used? (show work)

\[
82 - 2 \rightarrow 80 / 20 \rightarrow 4 = .8(k+1)/2 + .2 \times k
\]

\[
8 = .8k + .8 + .4k \rightarrow 7.2 = 1.2k \rightarrow k = \frac{7.2}{1.2} = 6
\]

\[
210/B = 6 \rightarrow B = 35
\]

new number of buckets: **35 buckets**

**Part C** Implement a function `findpos` that returns the earliest position of the character x in a character string. If x does not appear in the character string, return -1. For example:

```c
char sample[] = "Happy Birthday";
findpos('p', sample);    /* returns 2 */
findpos('m', sample);    /* returns -1 */
findpos('H', sample);    /* returns 0 */
```

```c
int findpos(char x, char *str) {
    int Pos;
    for (Pos = 0; str[Pos] != '\0'; Pos++) {
        if (str[Pos] == x)
            return(Pos);
    }
    return(-1);
}
```