CMSC 14300 - Summer 2025 — Konstantinos Ameranis
due 8/4/2025 via Gradescope

H5.

Problem 1 (Bad Hash functions) *Catherine (our grader) tried suggesting some alternative hash functions that should help with performance. The first one is $f_1(k) = 1$, which she claims is better since it is super fast.*

(a) (2 points) Using f_1 turns the hash table into which known data structure?

(b) (2 points) The second one she suggested was

```
uint64_t string_hash(void *key) {
    char *str = key;
    uint64_t hash = 0;
    int c;

    while ((c = *str++))
        hash = c + (hash << 6)

    return hash;
}
```

This one is a variation on the sdbm, but performs a single addition instead of three.

Given the string "information" construct an English word `s` *such that $f_2(\text{"information"}) == f_2(s)$. Is there a simple way to construct a collision for any input?*

(c) (2 points) Another way that Catherine is trying to help is by trying to make sure the hash function is truly random. Her proposal is $f_3(k) = hash(k)^r and int(2 * *64)$.
Why is this not a valid hash function?

Problem 2 (Timing comparisons (4 points)) *Compare the running time of* `groups` *using a hash table and a binary search tree, how long does it take to group the largest input in each case?*
Try:

```
/usr/bin/time ./groups -t tests/large.txt > /dev/null
/usr/bin/time ./groups -m tests/large.txt > /dev/null
```

There are primarily two parts in the `groups` *program:*

1. *insert all lines into the dictionary, and*
2. *visit all key-value pairs in sorted order*

Which implementation is better at accomplishing Step 1? Which implementation is better at Step 2?

Problem 3 *How many hours did you spend for this assignment?*

Problem 4 *What is the most difficult aspect of this assignment, if any?*

Problem 5 *Document your collaboration here.*