Solutions to Homework Nine

CSE 101

1. (a) Each time you roll a die, the chance of getting a six is $p = 1/6$. So the expected number of rolls until you see a six is $1/p = 6$.
   (b) Each time you pull out a random fish, the chance of getting a catfish is $p = 50/1000 = 1/20$. Thus the expected number of fish you need to pull out until you get a catfish is $1/p = 20$.
   (c) Each time the computer chooses a random integer, it has a probability $p = 1/10$ of getting a multiple of 10. Therefore the expected number of trials is $1/p = 10$.

2. (a) On any iteration, the probability of outputting `H` is $p(1-p)$ (probability of getting heads, then tails) and the probability of outputting `T` is $(1-p)p$ (probability of getting tails, then heads). Since these are equal, both outcomes are equally likely.
   (b) The probability that a particular iteration is successful (that is, the algorithm halts) is $2p(1-p)$. Thus the expected number of iterations is $1/(2p(1-p))$ and the expected number of coin tosses is twice this, $1/(p(1-p))$.

3. Here's the algorithm, given input x:

```
Repeat 100 times:
     Run A(x)
     If it says ``not prime'', output ``not prime'' and halt
Output ``prime''
```

If x is "prime", then $\mathcal{A}$ will always return "prime", and hence the answer will be correct. If x is not prime, then the probability that $\mathcal{A}(x)$ returns "prime" is at most $1/2$, and the probability that it returns "prime" 100 times is at most $1/2^{100}$.

4. (a) Here's an algorithm that makes use of both $\mathcal{A}$ and $\mathcal{B}$. On input x,

```
Run A(x)
If it says ``not prime'':  output ``not prime'' and halt
Run B(x) and output the answer
```

   To see why this works, suppose first that x is prime. Then $\mathcal{A}(x)$ will return "prime" and thus $\mathcal{B}(x)$ will be invoked, returning the right answer. On the other hand, if x is not prime, then either $\mathcal{A}(x)$ will detect this, or $\mathcal{B}(x)$ will be invoked.
   (b) If the input is prime, both procedures will be called, for a running time of $101T(n)$.
   (c) If the input is not prime, then $\mathcal{A}(x)$ will detect this with probability at least $1/2$; otherwise $\mathcal{B}(x)$ will also need to be called. Thus the expected running time is $T(n) + 0.5 \times 100T(n) = 51T(n)$.

5. This is equivalent to throwing n balls in n bins. The size of the largest bin is (with high probability) $O(\log n)$, and this is therefore the number of hours the repairs would take overall.

6. We saw in class that when n balls are thrown into n^2 bins, there is at least a $1/2$ probability that there will be no collisions. Therefore, we should set $2^m = n^2$, roughly, which means $m = 2\log n$.

7. *Hashing with open addressing.*
   (a) If n items are stored in a table of size $2n$, half the table is empty. Let's say we are inserting x. The locations $h(x,0), h(x,1), \ldots$ are random and thus each of them has at least a $1/2$ chance of being empty. The probability the first k locations are all occupied is therefore at most $1/2^k$.
   (b) Let A_i be the event that the ith insertion requires more than k probes.

$$\Pr(\text{some insertion requires more } k \text{ probes}) = \Pr(A_1 \cup A_2 \cup \cdots \cup A_n) \leq \Pr(A_1) + \cdots + \Pr(A_n) \leq \frac{n}{2^k}.$$

   For $k = 2\log_2 n$, this is $1/n$.

8. *Skip lists.*

   (a) Roughly speaking, a random element will on average lie halfway down the list and will thus need about $n/2$ time to locate. More precisely, the time taken to find the ith item in the list is i. For an element chosen at random, the expected lookup time is thus

   $$\sum_{i=1}^{n} \Pr(\text{item } i \text{ is chosen})\, i \;=\; \frac{1+2+\cdots+n}{n} \;=\; \frac{n+1}{2}.$$

   The worst-case lookup time is n.

   (b) Very roughly, a random element will on average be about halfway down the list, and will therefore be reached by about $n/(2k)$ jump pointers and $k/2$ next pointers, for a total time of $n/(2k) + k/2$. The worst-case lookup time is $n/k + k$.

   (c) The best choice is $k \approx \sqrt{n}$, leading to an expected and worst-case lookup time of $O(\sqrt{n})$.

9. We saw in class that when n items are stored in a Bloom filter of size m, using k hash functions, the fraction of zero entries in the table is roughly $e^{-kn/m}$. Therefore, a reasonable way to estimate n is as follows:

   - Determine the fraction of zero entries in the table; call this q.
   - Return $(m/k)\ln(1/q)$.

10. *The secretary problem.*

    (a) Let s_1 denote the best secretary and s_2 the second-best secretary. If s_2 is one of the first r candidates (which happens with probability r/n) and s_1 is one of the remaining $n-r$ candidates (which happens with probability $(n-r)/n$), then Barbara will correctly identify s_1.

    $$\Pr(\text{Barbara chooses } s_1) \geq \frac{r}{n} \cdot \frac{n-r}{n} = \frac{r(n-r)}{n^2}.$$

    (a) Setting $r = n/2$ results in a 1/4 probability of success.