Nombre: Javier Ignacio Gómez Peña
RUT: 19.516.104-6

P1 a)

Por el principio del palomar, podemos decir (a modo de ejemplo), que si tenemos k cajas y n elementos (con n > k), existirá al menos una caja con dos o más elementos.
Aplicando esta idea al problema, es importante notar que tanto a f como a g se les aplica módulo 100. Esto quiere decir que el mínimo valor que pueden tomar es 0 y el máximo es 99. Consideremos '100 cajas'. Para una cantidad mayor a 100 de elementos (n), necesariamente existirá un n_0 y n_1 tal que $f(n_0) = f(n_1)$ y un k_0, k_1tal que $g(k_0) = g(k_1)$. Por lo tanto, ambas funciones son periódicas cada 100 elementos.
Si analizamos las ecuaciones de recurrencia de ambas funciones, tenemos lo siguiente:
Para f:

$$f(n) = 5f(n-1) + 2g(n-1)$$
$$\Rightarrow f(n) = 5f(n-1) + 2g(n) - 6f(n-1)$$
$$\Rightarrow f(n) = -f(n-1) + 2g(n)$$
$$\Rightarrow g(n) = \frac{f(n) + f(n-1)}{2}$$

Entonces:

$$f(n) = 5f(n-1) + 2g(n-1)$$
$$\Rightarrow f(n) = 5f(n-1) + f(n-1) + f(n-2)$$
$$\Rightarrow f(n) = 6f(n-1) + f(n-2)$$

Luego, para g:

$$g(n) = 3f(n-1) + g(n-1)$$
$$\Rightarrow f(n-1) = \frac{g(n) - g(n-1)}{3}$$
$$\Rightarrow f(n) = \frac{5}{3}\big(g(n) - g(n-1)\big) + 2g(n-1)$$
$$\Rightarrow f(n) = \frac{5g(n) + g(n-1)}{3}$$

De la parte anterior (ecuación de recurrencia para f) teníamos que:

$$g(n) = \frac{f(n) + f(n-1)}{2}$$
$$\Rightarrow g(n) = \frac{5g(n) + g(n-1) + 5g(n-1) + g(n-2)}{6}$$
$$\Rightarrow g(n) = \frac{5g(n) + 6g(n-1) + g(n-2)}{6}$$
$$\Rightarrow g(n) = 6g(n-1) + g(n-2)$$

(se omitieron los módulos ya que no afectan en la ecuación de recurrencia)

Como podemos ver, f y g tienen la misma ecuación de recurrencia y, al tener el mismo módulo, tienen la misma periodicidad. Por lo tanto, $n_0 = k_0 = n$ y $n_1 = k\_1$ tal que $f(n_0) = f(n_1)$ y $g(n_0) = g(n_1)$. En otras palabras, ambas funciones tienen el mismo periodo a partir del mismo entero n.

P1 b)

Para calcular $\sum_{i=0}^{k}(f(i)+g(i))$ debemos calcular primero $f(i)$ y $g(i)$ por separado.
Para calcular $f(i)$ crearemos un arreglo de 100 elementos donde almacenamos los elementos para $f(i)$ desde i = 0 hasta i = 99 (los cuales calculamos recursivamente con la función entregada en el enunciado). Sabemos que la función es periódica cada 100 elementos, por lo que para calcular $f(100)$ basta con consultar por $f(0)$, para $f(101)$ consultamos por $f(1)$, $para\ f(102)$ consultamos por $f(2)$, y así consecutivamente (en otras palabras, $f(i) = f(i\%100)$).
Análogamente, hacemos lo mismo para g. Creamos un arreglo de 100 elementos donde tendremos los valores para $g(i)$ desde i = 0 hasta i = 99. Luego, podemos tener el valor para $f(i)$ y $g(i)$ en tiempo lineal, y con esto, podemos tener el valor de $f(i) + g(i)$.
Luego, hacemos la suma sobre i desde 0 hasta k de $f(i) + g(i)$.
El costo de realizar las consultas y la suma es lineal en k, pero debemos sumarle el costo de calcular $f(i)$ y $g(i)$ de 0 hasta 99, pero este costo es constante para k, ya que siempre calculamos hasta $i = 99$, independiente del valor de k.
Por lo tanto, este algoritmo es $O(k)$.

P1 c)

Archivo p1c.txt (Python 2.7)

P1 d)

Cómo vimos en la parte a), f y g poseen la misma ecuación de recurrencia, la cual no depende del módulo.
Por el mismo argumento utilizado en la parte a), si f tiene módulo m, sólo puede tomar valores entre 0 y $m-1$. Lo mismo para g; si tiene módulo l, sólo puede tomar valores entre 0 y $l-1$. Luego f será m-periódica y g será l-periódica.

P1 e)

Archivo p1e.txt (Python 2.7)

P2 a)

Si consideramos la probabilidad de ganarle a un jugador aleatorio, podemos decir que es equivalente al producto entre la probabilidad de ganarle a un jugador dado y la probabilidad de elegir a ese jugador (por principio multiplicativo).
En primer lugar, la probabilidad de elegir un jugador aleatorio X es:

$$\frac{1}{J}$$

siendo J la cantidad total de jugadores, excluyendo al principal.

Luego, la probabilidad de ganarle a un jugador 'J', que pertenece a un grupo 'i' es:

$$\frac{h_x}{h_x + h_i}$$

Luego, la probabilidad de ganarle a un jugador aleatorio es:

$$\frac{1}{J} \cdot \sum_{i=1}^{k} G_i \cdot \frac{h_x}{h_x + h_i}$$

siendo k la cantidad total de grupos, y G_i la cantidad total de jugadores pertenecientes al grupo 'i'. Este producto es debido al principio multiplicativo de las probabilidades.

P2 b)

Archivo p2b.txt (Python 2.7)

P2 c)

Para que el jugador gane el premio, debe ganar una determinada cantidad de partidas que en total le entreguen una cantidad n de puntos. El jugador dejará de jugar una vez que haya alcanzado los n puntos y con esto, ganado el premio.
Cabe destacar que la probabilidad de que el jugador gane una partida cualquiera es la calculada en la parte a)

$$p = \frac{1}{J} \cdot \sum_{i=1}^{k} G_i \cdot \frac{h_x}{h_x + h_i}$$

Tenemos dos posibles casos:

1. El jugador gana n partidas seguidas sin haber perdido ninguna (nunca pierde puntos). En este caso la probabilidad de que gane el premio es p^n

2. El jugador en algún momento pierde puntos. En este caso, si consideramos que debe ganar i partidas para obtener los n puntos, entonces se puede inferir que debió haber perdido $i - n$ partidas. Entonces el total de partidas que debe jugar es

$$2i - n$$

Luego, sabemos que el total de partidas que se jugarán es m.
Si $2i - n > m$, entonces la probabilidad de que gane el premio es 0, puesto que no alcanzaría los n puntos.
Si $2i - n < m$, entonces el jugador tiene posibilidad de ganar el premio.
Definamos una variable aleatoria X como el número de partidas que el jugador ganó.
Es fácil ver que nuestra variable aleatoria X tiene una distribución binomial, por lo que la probabilidad de que $X = k$ es:

$$P(X = i) = \binom{2i - n}{i} \cdot p^i \cdot (1 - p)^{i-n}$$

Es importante notar que nuestra variable aleatoria considera el caso en que el jugador gane n veces seguidas al principio, independientemente del resultado de las partidas restantes (recordemos que $i > n$, y $m = 2i - n$), por lo que debemos quitarle esos casos.
Para esto, primero debemos contar todos los casos en que se gane n veces seguidas al comienzo. Sabemos que se ganarán i partidas, pero si consideramos que ya se ganaron n, nos quedan $i - n$ partidas por ganar, las cuales se deben distribuir en el número de partidas restantes, el cual corresponde a $2i - n - n$ (ya que sabemos que se jugaron n partidas las cuales se ganaron todas). Por lo tanto, existen

$$\binom{2(i - n)}{i - n}$$

formas distintas de distribuir las partidas que quedan por ganar en las partidas restantes.
Entonces, debemos restarle estas combinaciones ya que el caso en que gane n partidas seguidas al comienzo ya está siendo contado cuando tenemos p^n.
Luego, debemos realizar la suma de todos los casos:

- $i = n$, donde la probabilidad de ganar es $P = p^n$
- Para todos los i desde $n + 1$ hasta m (sabemos que $i > n$)

Nos queda que la probabilidad de ganar es:

$$P = p^n + \sum_{i=n+1}^{m} \left(\binom{2i - n}{i} - \binom{2(i - n)}{i - n} \right) \cdot p^i \cdot (1 - p)^{i-n}$$

Con m la cantidad de partidas jugadas, n la cantidad de puntos necesarios para ganar el premio y p la probabilidad de ganar un partido aleatorio (parte a)).

Luego, el algoritmo para calcular esta probabilidad sigue los siguientes pasos:

1. Conocer n, m, h_x, J, k, G_i y h_i, datos necesarios del problema (indicados en el enunciado)

2. Calcular p, la probabilidad de ganar un partido aleatorio (parte a)
3. Calcular P la probabilidad de ganar el premio, según la formula enunciada anteriormente
4. Entregar la probabilidad P de ganar el premio

P2 d)

Archivo p2d.txt (Python 2.7)