```
import random

def simulate_hashing(b, n, trials):
   no_collisions = 0

   for x in range(trials):
      hash_name = [random.randint(1, b) for y in range(n)]
      if len(set(hash_name)) == n:
         no_collisions += 1

   probability = float(no_collisions / trials)

   return probability


buckets = 10000
trials = 1000
largest_n = 115

for n in range(115, 126):
   probability = simulate_hashing(buckets, n, trials)
   print(f"n = {n}: P(X) = {probability}")
   if probability > 0.5:
      largest_n = n

print(f"\nThe largest value of n such that the probability is greater than 0.5 is: {largest_n}")
```

0: Hashing

(a) Let X be the event that no two picks hash into the same bucket. Let n be the number of names chosen from a universe of N names and b be the number of buckets. We are given that the hash function maps exactly N/b names into the b buckets. We are also given that we are choosing n names with replacement. Let's consider the first pick. Since there are no names that have been hashed, this name can be hashed anywhere without collision. Therefore, the probability that this name gets hashed without collision is $\frac{b}{b} = 1$. For the second pick, we want to avoid one of the buckets, leaving $b-1$ buckets. Thus, the probability that this name gets hashed without collision is $\frac{b-1}{b}$. We can continue this for all n buckets leaving us with $P(X) = \frac{b}{b} \cdot \frac{b-1}{b} \cdot \frac{b-2}{b} \cdot ... \cdot \frac{b-(n-1)}{b} = \frac{b!}{b^n(b-n)!}$.

(b) Output:

```
n = 115: P(X) = 0.517
n = 116: P(X) = 0.497
n = 117: P(X) = 0.506
n = 118: P(X) = 0.502
n = 119: P(X) = 0.474
n = 120: P(X) = 0.485
n = 121: P(X) = 0.488
n = 122: P(X) = 0.466
n = 123: P(X) = 0.467
n = 124: P(X) = 0.442
n = 125: P(X) = 0.439
The largest value of n such that the probability is greater than 0.5 is: 118
```

(c)

1: Tournament Champions

(a) As a graph, the players would be the vertices. Each edge would be the game played between the players/vertices that it connects. The edges would be directed from the winner to the loser. Since there is no other information that is needed, the graph would not be weighted.

(b) We will prove this by induction.

Claim: There exists a "champion" player that beats every other player, or there is a 3-cycle.
Base Case: n = 2 By the definition of a 2-player tournament, there must exist a player that beats another player. Therefore, there exists a "champion".
Induction Hypothesis: There exists a "champion" player that beats every other player, or there is a 3-cycle for an n-player tournament.
Inductive Step: Let us consider a tournament with n+1 players. If there is a champion player that beats every player, then we are done. Therefore, let us assume that every player is beaten by some other player in the tournament. If we remove one player, we are left with a tournament with n players. We are left with two cases:

(1) Every player in the tournament is beaten by some other player.
(2) There exists a "champion" player.

In the first case, if every player is beaten by some other player, then by the Induction Hypothesis, we have a 3-cycle in the n-player tournament. The 3-cycle will also exist in the tournament with n+1 players so in this case the inductive step holds. In the second case there is a champion of the n-player tournament. However, since we originally established that every player is beaten by some other player in the n+1-player tournament, then the player originally left out of the n-player tournament must beat the champion. Furthermore, since this new champion must also lose to some player, it follows that this player is part of the n-player tournament, thus creating a 3-cycle.

Since both the base case and inductive step hold, we have proven that either a "champion" player or a 3-cycle exists for every tournament.

(c) By definition of a tournament, any subset of 2 or more of its players is also considered a tournament. Additionally, if any sub-tournament has no consistent ranking it follows that the entire tournament also has no consistent ranking. Therefore, if we prove that if every subset of three players has a consistent ranking, we can conclude that the entire tournament also has a consistent ranking. We will prove this by induction.

Claim: A tournament has no consistent ranking iff some subset of three of its players has no consistent ranking.
Base Case: $\mathbf{n = 2}$ Every two player tournament has a consistent ranking.
Induction Hypothesis: An n-player tournament has a consistent ranking if all subsets of three players have consistent rankings.
Inductive Step: Let us consider a tournament with n+1 players in which all subsets of three players have a consistent ranking. It follows that the tournament cannot have a 3-cycle since there is no consistent ranking in a 3-cycle. Thus, by part (b), there must exist a champion in this tournament. If we remove this champion from the n+1 players, it follows that the remaining n-player tournament also has no 3-cycles. By the Induction Hypothesis, it follows that this tournament has a consistent ranking. If we take this ranking and add the removed champion player, we are left with a consistent ranking of the entire n+1-player tournament.

Thus, we have proven that a tournament has no consistent ranking iff some subset of three of its players has no consistent ranking.

2: Six Color Theorem

(a) We are given Euler's Formula that for every connected planar graph with v vertices, e edges, and f faces, that $v - e + f = 2$. Rearranging this equation, we have $v = e - f + 2$. To show that

(b)

(c)

3: Average BST Height

(a) Insight 1 must be true due to the order in which elements are added to a Binary Search Tree. Let us consider an element x and its search path in the constructed tree. If i, where $i < x$, is on the search path for x, it means that i must have been inserted before x. Additionally, i being on the search path for x, it implies that i was encountered before x during the insertion process, and the resulting tree would reflect this. Thus, Insight 1 must be true.

(b) If we let X_i be an indicator random variable for the event that i is on the search path for x, then $X_i = 0$ if i is not on the search path and $X_i = 1$ if i is on the search path. Thus, the expected value for X_i is given by $\mathbb{E}[X_i] = P(X_i = 1) \cdot 1 + P(X_i = 0) \cdot 0 = P(X_i = 1)$. Therefore, we know that the expected value for X_i is only dependent on the probability that i is on the search path for x. From Insight 1, we know that this happens iff i is the first element added from the set $\{i, i+1, ..., x-1, x\}$. There are a total of $x - i + 1$ elements in this set and since we are assuming a random permutation of [N], any of these elements can be added first with the same probability. Thus, the expected value is $\mathbb{E}[X_i] = \frac{1}{x-i+1}$.

(c) $\mathbb{E}[X]$ is the expected length of the search path for an arbitrary element x in the tree. If we let X_i be an indicator random variable as defined in part (b), then the expected length of the search path is equal to the sum of the indicator variables for all i from 1 to x.

$$\begin{aligned}
\mathbb{E}[X] &= E[X_1 + X_2 + ... + X_x] && \text{Expected Value} \\
&= E[X_1] + E[X_2] + ... + E[X_x] && \text{Linearity of Expectation} \\
&= \frac{1}{x-1+1} + \frac{1}{x-2+1} + ... + \frac{1}{x-x+1} && \text{Formula from (b)} \\
&= \frac{1}{x} + \frac{1}{x-1} + ... + \frac{1}{1} && \text{Simplification}
\end{aligned}$$

This is the harmonic series. Therefore, $\mathbb{E}[X] = H_n$.

(d)

i.

We can relate the integral of $\frac{1}{x}$ to the summation of $\frac{1}{x}$ to bound H_n. The integral of a function, $f(x)$, from a to b can be approximated by the sum of the areas of rectangles under the curve. If we use left-endpoint rectangles, we can evaluate the function at the left endpoint of each subinterval and use the height of the rectangle to approximate the area. Since we know that $\ln(x) = \int \frac{1}{x}$, we can substitute the integral into the inequality: $\int \frac{1}{x} < H_n < \int \frac{1}{x} + 1$.

For the integral, $\int_1^n \frac{1}{x} dx$, The subintervals are in the form [k-1, k] for k from 1 to n.
Left Endpoint (Lower Bound):
For the left-endpoint rectangles, we evaluate the function $\frac{1}{x}$ at the left endpoint of each subinterval. The left endpoint of each subinterval is $x = k-1$. The approximation using left-endpoint rectangles is $\int_1^n \frac{1}{x} dx \approx \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n-1}$. This approximation is equal to H_{n-1} which we know is less than H_n.
Right Endpoint (Upper Bound):
For the right-endpoint rectangles, we evaluate the function $\frac{1}{x}$ at the right endpoint of each subinterval. The left endpoint of each subinterval is $x = k$. The approximation using left-endpoint rectangles is $\int_1^n \frac{1}{x} dx \approx \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$. This approximation is equal to H_n, so if we add 1 to the right side, it follows that $H_n < H_n + 1 = \ln(n) + 1$.

Thus, we have shown that $\ln(n) < H_n < \ln(n) + 1$.

ii.

We have shown previously that $\mathbb{E}[X] = H_n$ and that $\ln(x) < H_n < \ln(x) + 1$. Therefore, we can use the upper bound of this to show that $\mathbb{E}[X] \leq a \cdot \lg(N) + b$ for some $a \in \mathbb{Q}^+$ and $b \in \mathbb{Z}$. This means that using $H_n < \ln(n) + 1$, we can manipulate the right side of the equation to be in the form $a \cdot \lg(N) + b$. If we have $\ln(n) + 1 = a \cdot \lg(N) + b$, its clear that $b = 1$. To find a, we can use change of base in logs to say that $\lg(N) = \frac{\ln(N)}{\ln(2)}$. Thus, we know that $a \cdot \lg(N) = a \cdot \frac{\ln(N)}{\ln(2)} = \ln(N)$. It follows that $a = \ln(2)$ which is in the realm of positive rational numbers. Thus, we have found an $a \in \mathbb{Q}^+$ and $b \in \mathbb{Z}$ such that $\mathbb{E}[X] \leq a \cdot \lg(N) + b$.