Name: Richard Duncan

Problem	Points	Score
1a	10	
1b	10	
1c	10	
1d	10	
2a	10	
2b	10	
3a	10	
3b	10	
3c	10	
3d	10	
Total	100	

Notes:

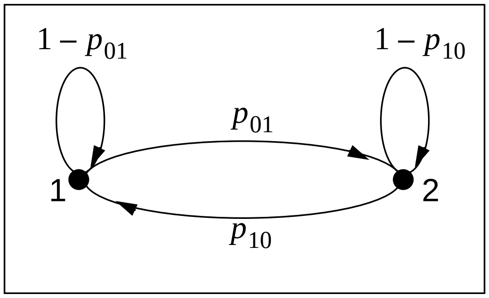
1. The exam is closed books/closed notes - except for one page (double-sided) of notes.

2. Please show ALL work. Answers with no supporting explanations or work will be given no credit.

3. Please indicate clearly your answer to the problem. If I can't read it (and I am the judge of legibility), it is wrong. If I can't follow your solution (and I get lost easily), it is wrong. All things being equal, neat and legible work will get the higher grade:)

Problem No. 1: Entropy Rate

(a) For the two-state Markov chain with the transition matrix, $P = \begin{bmatrix} 1-p_{01} & p_{01} \\ p_{10} & 1-p_{10} \end{bmatrix}$**, find the entropy rate.**



A graphical representation of this Markov chain is shown at right. The stationary distribution, μ, can be found by solving the equation $\mu P = \mu$, which balances the probabilities, leading us to the equation $\mu_1 p_{01} = \mu_2 p_{10}$. Since this is a pdf, $\mu_1 + \mu_2 = 1$, hence

$\mu_1 = \dfrac{p_{10}}{p_{01}+p_{10}}$ and $\mu_2 = \dfrac{p_{01}}{p_{01}+p_{10}}$.

The entropy rate of a stationary Markov chain is $H = -\sum_{i,j} \mu_i P_{i,j} \log P_{i,j}$.

For our two state Markov chain, this becomes $H = \dfrac{p_{10}}{p_{01}+p_{10}} H_0(p_{01}) + \dfrac{p_{01}}{p_{01}+p_{10}} H_0(p_{10})$, where $H_0(p) = -p\log p - (1-p)\log(1-p)$ is the entropy of a binary distribution.

(b) Find the values of p_{01} **and** p_{10} **that maximize the entropy rate.**

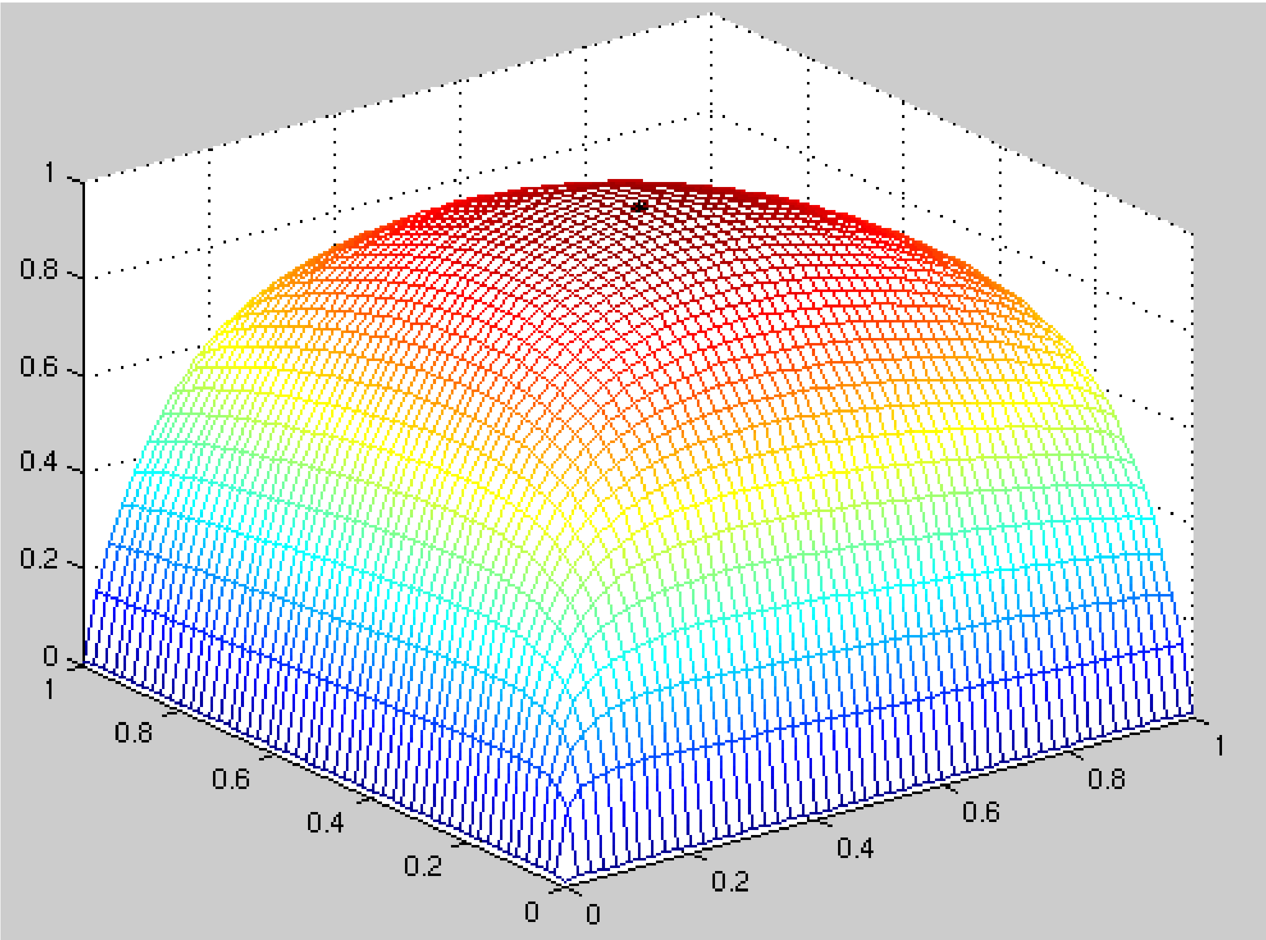
This could be a tricky problem since we have to optimize two parameters simultaneously. One approach would be to first maximize the binary distributions $H_0(p_{01})$ and $H_0(p_{10})$. The maximum entropy for each is obviously the uniform distribution, so $p_{01} = p_{10} = 0.5$.

This also yields the stationary distribution will be uniform, as $\mu_1 = \mu_2 = \dfrac{0.5}{0.5+0.5} = 0.5$. The entropy rate for this system will now be:

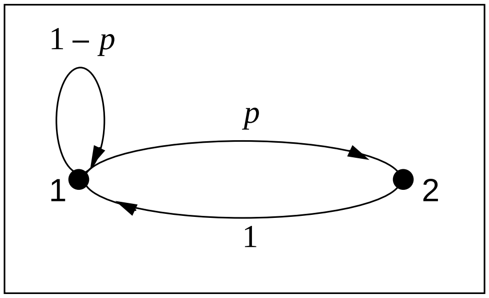
$$H = \frac{p_{10}}{p_{01}+p_{10}} H_0(p_{01}) + \frac{p_{01}}{p_{01}+p_{10}} H_0(p_{10}) = (0.5)(1\text{bit}) + (0.5)(1\text{bit}) = 1\text{bit}$$

This can be numerically verified through Matlab. The following matlab code produces a three dimensional plot. The x and y axis are p_{10} and p_{01}, respectively, while the vertical axis is the resulting entropy rate. A black star is placed at my estimated maximum entropy point, 0.5 for both parameters, and this point does correspond to the maximum peak of the graph at 1 bit. The graph is consistent with our knowledge that entropy is a concave function.

```
clear; close; syms a b h ha hb

ha = -a * log(a)/log(2.0) - (1-a) * log (1-a) / log(2.0);
hb = -b * log(b)/log(2.0) - (1-b) * log (1-b) / log(2.0);
h = (b/(a+b)) * ha + (a/(a+b)) * hb;

npoints = 51;
points = linspace(0.001,0.999,npoints);

for i = 1:npoints
    a = points(i);
    for j = 1:npoints
        b = points(j);
        h1(i,j) = eval(h);
    end
end

mesh(points,points,h1);
shading interp;
colormap(jet);
hold
plot3(0.5,0.5,1,'k*')
```



(c) For the two-state Markov chain with the transition matrix, $P = \begin{bmatrix} 1-p & p \\ 1 & 0 \end{bmatrix}$**, find the maximum value of the entropy rate.**



This problem is just a simplified version of the previous. The graph of this Markov chain is shown at right. The main difference is that the second state is now completely deterministic: once entered it always transitions back to the first state. Recalling our equation for the system's entropy rate from part (a),

$$H = \frac{p_{10}}{p_{01}+p_{10}}H_0(p_{01}) + \frac{p_{01}}{p_{01}+p_{10}}H_0(p_{10}),$$

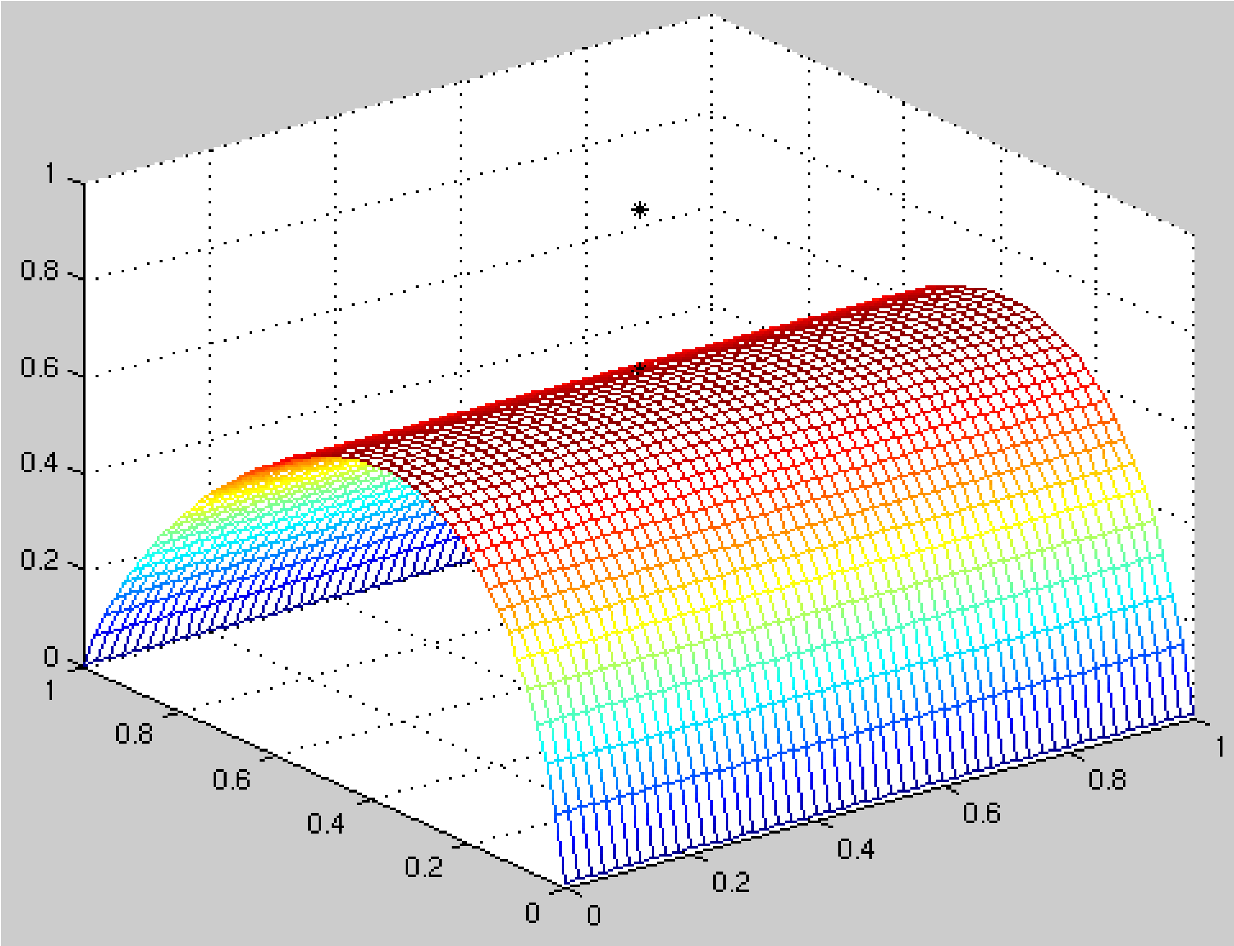
Substituting in $p_{10} = 1$ and $p_{01} = p$, we obtain

$H = \frac{1}{p+1}H_0(p) + \frac{p}{p+1}H_0(1) = \frac{1}{p+1}H_0(p)$. Since I don't like to take derivatives, I will again guess that a uniform distribution will produce the greatest entropy and confirm my guess with visualization. If we use the uniform distribution for p, we obtain

$H = \frac{1}{0.5+1}1\text{bit} = 0.6667\text{bits}$. Reusing the same matlab code with a few modifications,

```
clear;close;syms a b h ha hb

ha = -a * log(a)/log(2.0) - (1-a) * log (1-a) / log(2.0);
hb = -b * log(b)/log(2.0) - (1-b) * log (1-b) / log(2.0);

h = (b/(a+b)) * ha + (a/(a+b)) * hb;

npoints = 51;points = linspace(0.001,0.999,npoints);

for i = 1:npoints
    a = points(i);
    for j = 1:npoints
        b = 0.99999;
        h1(i,j) = eval(h);
    end
end

mesh(points,points,h1);
shading interp;
colormap(jet);
hold
plot3(0.5,0.5,1,'k*')
plot3(0.5,0.5,2/3,'k*')
```

The main difference is that now there is only variance along one axis, the other is fixed. I place a star at the maximum entropy point for both Markov chains, the one along the curve is at the expected point of $p = 0.5$ and $H = 0.6667$. The plot is shown on the next page.



(d) Explain the reasons for any differences between the answers to (b) and (c), and any significance to this result. Your answer must show insight and understanding — don't simply tell me that the numbers are different because the probabilities are different.

The entropy rate of a Markov process can be considered to be the expected entropy of the system. What I mean by this is that it is the average of each state's entropy weighted by the expected percentage of time that the system will be in that state. For the first model where we have freedom in the parameters for both states, we can assign both states maximum entropy. In this case, the weighting function is irrelevant since both distributions have the same entropy (although it is easy to show that the system is equally likely to be found in either state). So, for the first model the entropy rate will simply be the entropy of one of the state's pdf's, or 1 bit.

For the second model, however, we only have freedom in the first state. Again, we assign maximum entropy to this state to maximize the overall entropy rate, but the second state's determinism ends up decreasing the weighted average. To find the weighting function, consider how long the system will stay in each state. For the first state, there is an equal chance of staying in the same state of transitioning to the second state, while the second state always transitions back to the first state. Due to the chance of staying in state one, the system is twice as likely to be found in state 1 as state 2, hence the 0.66, 0.33 distribution. The second state has no uncertainty, while the first state has maximum entropy for a binary distribution (1 bit). By conditioning the state entropies by the weighting function we arrive at the overall entropy rate of 0.6667 bits.

Problem No. 2: Data Compression

(a) Given an alphabet $S = \{S_1, S_2, \ldots, S_6\}$, and associated probabilities $P = \{p_1, p_2, \ldots, p_6\}$, you design an optimal D-ary code whose lengths turn out to be $L = \{1, 1, 2, 3, 2, 3\}$. Find a lower bound on D.

In order to compress data, we must minimize the expected codeword length. This means that the more probable symbols have shorter codewords. Since $l_1 \le l_2 \le l_3 \le l_5 \le l_4 \le l_6$, it must follow that $p_1 \ge p_2 \ge p_3 \ge p_5 \ge p_4 \ge p_6$. This is not vital information for solving the problem. Since there are few codewords, we will first try $D = 2$ and then try $D = 3$.

Any optimal code can be arranged as a prefix code. In trying to construct a prefix code which satisfies L, it becomes obvious very early that $D > 2$. This is because there are two codewords of length one, **1** and **0**. Obviously any other codeword in a binary code must start with a **1** or a **0**, so the resulting code cannot be a prefix code. Since any optimal code can be arranged as a prefix code and we proved that no such prefix code may exist, then no such optimal code can exist with $D = 2$.

Since $D = 2$ doesn't work, let's try $D = 3$. The two one-length codewords can be assigned **0** and **1**, leaving **2** to prefix the other codewords. Similarly, assign **20** and **21** to be the two two-length codewords, leaving **220** and **221** for the final codewords of length three. Since $D = 3$ works, (and $D = 2$ doesn't), the lower bound on D is 3.

This can be shown in a more mathematically rigorous fashion through the Kraft inequality. It states that for any uniquely decodable code the codeword lengths must satisfy $\sum D^{-l_i} \le 1$. So we must find the lower bound on D such that $D^{-1} + D^{-1} + D^{-2} + D^{-3} + D^{-2} + D^{-3} \le 1$. This simplifies to $D^{-1} + D^{-2} + D^{-3} \le 1/2$. Since I begin to shudder at the thought of using Lagrange multipliers, instead I will just try a couple of numbers again, starting at two. $D = 2$ does not satisfy this, as $1/2 + 1/4 + 1/16 > 1/2$. $D = 3$ will satisfy this constraint, as $1/3 + 1/9 + 1/27 = 0.48 \le 0.5$. Therefore 3 is the lower bound (and Martians may have 3 fingers). In actuality, my method is just as valid as using calculus, since any number found through real analysis must then be converted into an integer anyway.

(b) Let X be a binary-valued random variable with probabilities {ε,1-ε}. Estimate the expected length of an optimal code and discuss how this relates to the source coding theorem.

The expected length of a binary code C can be expressed as $L(C) = \sum_{x \in X} p(x)l(x)$. To find an optimal code we must minimize this expected length, $\varepsilon l_1 + (1-\varepsilon)l_2$ under the constraint of the Kraft inequality, $2^{-l_1} + 2^{-l_2} \le 1$. Cover uses calculus and Lagrange multipliers to take these two equations to produce the equation $l_i^* = -\log p_i$ for non-integer optimal codeword lengths. So, in our case $l_1 = -\log\varepsilon$ and $l_2 = -\log(1-\varepsilon)$. If we were free to use non-integer values for codeword lengths (maybe that is how they do it on Mars), our expected codeword length will be $\varepsilon(-\log\varepsilon) + (1-\varepsilon)(-\log(1-\varepsilon))$. This expected codeword length is clearly the entropy of the binary random variable $H_0(\varepsilon)$.

The source coding theorem states that the expected length of the optimal code will be bounded by entropy and entropy plus 1, $H(X) \le L^* \le H(X) + 1$. Since the best that our optimal code can possibly due is entropy (for a dyadic distribution), the lower bound has already been shown to hold true. In order to show that we also meet the upper bound criterion, we must revisit the Kraft inequality, this time using the ceiling function on our codeword lengths to force an integer restraint on the values. Now, $l_1 = \lceil -\log\varepsilon \rceil$ and $l_2 = \lceil -\log(1-\varepsilon) \rceil$, so the inequality becomes

$$2^{-\left\lceil \log\frac{1}{\varepsilon} \right\rceil} + 2^{-\left\lceil \log\frac{1}{1-\varepsilon} \right\rceil} \le 2^{-\log\frac{1}{\varepsilon}} + 2^{-\log\frac{1}{1-\varepsilon}} = \varepsilon + (1-\varepsilon) \le 1$$

The first inequality is true because the ceiling function always increases the negative exponent, resulting in smaller terms for the sum than the non integer constrained sum. Since the ceiling function will always add a number less than one, $\log\frac{1}{\varepsilon} \le l_1 \le \log\frac{1}{\varepsilon} + 1$, hence the upper bound on the codeword length may be expressed as

$$L^* \le \varepsilon\left(\log\frac{1}{\varepsilon} + 1\right) + (1-\varepsilon)\left(\log\frac{1}{1-\varepsilon} + 1\right) = \qquad ,$$
$$\varepsilon\log\frac{1}{\varepsilon} + (1-\varepsilon)\log\frac{1}{1-\varepsilon} + \varepsilon + (1-\varepsilon) = H_0(\varepsilon) + 1$$

which provides our upper bound. So, our decision to use codeword lengths $\{\left\lceil \log\frac{1}{\varepsilon} \right\rceil, \left\lceil \log\frac{1}{1-\varepsilon} \right\rceil\}$ is consistent with the source coding theorem.

Problem No. 3: Gambling and Complexity

Consider a three-horse race with probabilities $(p_1, p_2, p_3) = (1/2, 1/4, 1/4)$. The odds associated with this race are $(r_1, r_2, r_3) = (1/4, 1/4, 1/2)$ (recall, these are typically set by how people bet on the race, not what the underlying odds are). The race is run many times, and number of the winning horse is transmitted over a communications channel using an optimal compression scheme.

(a) How many bits are required to transmit this information?

The minimum number of bits required to transmit this information will simply be the entropy of the true pdf, $H(p) = -\left[\frac{1}{2}\log\frac{1}{2} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{4}\log\frac{1}{4}\right] = 1.5\text{bits/race}$

(b) Define an optimal betting strategy, (b_1, b_2, b_3), so that your wealth will increase as quickly as possible.

The optimal betting strategy will always be to set $b^* = p$. We can actually make money on this race since we are closer to the actual distribution than the bookie is. So, set $b = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$

We will make $ES = E(b \cdot o) = \sum_{k=1}^{3} p_k \log b_k r_k$, where S is our wealth and r is the bookies payback. Therefore our doubling rate will be:

$$W(b, p) = \frac{1}{2}\log\frac{4}{2} + \frac{1}{4}\log\frac{4}{4} + \frac{1}{4}\log\frac{2}{4} = 0.5 + 0 - 0.25 = 0.25.$$

This is also the relative entropy of the actual pdf to the bookie's pdf minus the relative entropy of the actual pdf to our pdf.

This means that after n games we should have $2^{n(0.25)}$ growth of our gambling investment.

(c) Is there only one solution to this problem? Explain.

There is only one *optimal* betting strategy. Since we know the actual probability distribution of the races. Therefore the optimal betting strategy is to fully utilize the actual pdf to the best of our advantage, distributing our money according to the pdf. Any other betting strategy will yield a smaller return.

If we are interested in any betting strategy which allows us to make money, there are many such distributions. Using the theorem that $W(b, p) = D(p||r) - D(p||b)$, we can solve for any betting strategy b which yields a positive $W(b, p)$. The bound on this is how far off the bookie is,

$$D(p||r) = \sum_{k=1}^{3} p_k \log\frac{p_k}{r_k} = \frac{1}{2}\log\frac{1/2}{1/4} + \frac{1}{4}\log\frac{1/4}{1/4} + \frac{1}{4}\log\frac{1/4}{1/2} = \frac{1}{2} + 0 - \frac{1}{4} = \frac{1}{4}.$$

So, for any betting strategy b, our doubling rate will be $W(b, p) = \frac{1}{4} - D(p||b)$.

This means that any distribution we choose that has a Kullback Leibler distance less than 0.25 from the actual distribution will increase our wealth over time. Since relative entropy is a non negative function, the optimal betting strategy will be when this distance is zero, or $b = p$.

(d) Two three-horse races with different probabilities and odds are run, and the results are again reported over the communications link. For example, we transmit a pair of numbers, (a, b), indicating that the first race was won by horse a, and the second by horse b. Hence, you observe data such as $\{(1, 1), (2, 3), (1, 3), \ldots\}$. Estimate the Kolmogorov complexity of this sequence of numbers in terms of the Kolmogorov complexity of each race reported individually over the same link. In other words, how does the two-event process compare to two one-event processes?

Let $K(a)$ be the Kolmogorov complexity of the string "horse a won the race." Let $K(a, n)$ be the complexity of the string "horse a won the nth race." It is clear that $K(a, n) = K(a) + c$, where c is the number of bits to create the race number, $\log(n+1)\text{bits}$. Therefore, to transmit two races will require $2(K(a) + c)$ bits.

If instead we combine the two races into a single string, we get something like "For two races, first horses a won and then horse b won. In this case we have $K(a, b) = K'(a) + K'(b) + c'$, where $K'(a)$ is the complexity of the string "first horse a" and $K'(b)$ is the complexity of the string "then horse b" and c' is the complexity of the surrounding text "For two races, _ won and _ won." If we let $K'(A)$ be the maximum over $K'(a)$ and $K'(b)$, we can generalize this to n races with $K'_n \le nK'(A) + c'$. This becomes a more profound savings as n increases, since we don't have to fully describe the situation for each race.