13.10 Deciding to put probability theory to good use, we encounter a slot machine with three independent wheels, each producing one of the four symbols BAR, BELL, LEMON, or CHERRY with equal probability. The slot machine has the following payout scheme for a bet of 1 coin (where "?" denotes that we don't care what comes up for that wheel):

- **BAR/BAR/BAR pays 20 coins**
- **BELL/BELL/BELL pays 15 coins**
- **LEMON/LEMON/LEMON pays 5 coins**
- **CHERRY/CHERRY/CHERRY pays 3 coins**
- **CHERRY/CHERRY/? pays 2 coins**
- **CHERRY/?/? pays 1 coin**

a. Compute the expected "payback" percentage of the machine. In other words, for each coin played, what is the expected coin return?

The possible worlds can be constructed using *permutation with repetition:*

Total Cases: $4^3 = 64$

The table below presents all different combinations, how much every combination pays, the number of cases of the combination in the space world and the expected global pay over 64 plays.

Permutations	Pays Per Case	Cases	Pays
BAR/BAR/BAR	20	1	20
BELL/BELL/BELL	15	1	15
LEMON/LEMON/LEMON	5	1	5
CHERRY/CHERRY/CHERRY	3	1	3
CHERRY/CHERRY/?	2	3	6
CHERRY/?/?	1	12	12
Rest of permutations	0	45	0
Total		**64**	**61**

$Payback = 61/64 \approx 95.3125\%$

A similar way to arrive to the same solution is making use the the probability of an atomic event:

Payback = P(BAR/BAR/BAR) x 20 + P(BELL/BELL/BELL) x 15 + P(LEMON/LEMON/LEMON) x 5 + P(CHERRY/CHERRY/CHERRY) x 3 + (P(CHERRY/CHERRY/?) - P(CHERRY/CHERRY/CHERRY)) x 2 + (P(CHERRY/?/?) - P(CHERRY/CHERRY/?)) x 1 = 20/64 + 15/64 + 5/64 + 3/64 + 6/64 + 12/64 = 61/64

b. Compute the probability that playing the slot machine once will result in a win.

$P(\textit{win in one shot}) = \textit{Winning Cases} / \textit{Total Cases} = 19/64 \approx 0.296875$

c. Estimate the mean and median number of plays you can expect to make until you go broke, if you start with 10 coins. You can run a simulation to estimate this, rather than trying to compute an exact answer.

```
import random

def slot(verbose=False):
    choice = [random.choice(['BAR','BELL','LEMON','CHERRY']) for i in range(3)]
    if verbose:
        print(choice)
    if choice[0] == choice[1] == choice[2]:
        if choice[0] == 'BAR':
            return 20
        elif choice[0] == 'BELL':
            return 15
        elif choice[0] == 'LEMON':
            return 5
        else: # CHERRY!
            return 3
    elif choice[0] == 'CHERRY':
        if choice[1] == 'CHERRY':
            return 2
        else:
            return 1
    else:
        return 0

def play():
    coins = 10
    plays = 0
    while coins > 0:
        coins -= 1
        coins += slot()
        plays += 1

    return plays


# run simulation
N = 10000
history = [play() for i in range(N)]
print(sum(history)/N)
history = sorted(history)
print(history[int(N/2)])
```

Results obtained in one simulation:

Mean: *197.3293*
Median: *21*