Problem 1.

We have the Time Epochs $t = 1, 2 \ldots, T$ (infinite-horizon problems). Let A be the result of this game, the state space $S = \{win, lose, draw\}$, the action space include two ways of playing $A = \{1,2\}$, 1 means play timid, and 2 means play bold. If the state is win, we give the reward 1. If the state is lose, we give the reward -1, and if the state is draw, we give the reward 0. We should let the sum of all reward $G_t = R_1 + R_2 + R_3 + \cdots + R_t$ be 1, when $t \geq 5$.

The transition probability is:

$$p(s'|s,a) \equiv \Pr(S_{t+1} = s'|S_t = s, A_t = a) = \sum_{r \in R} p(s', r|s, a)$$

And the reward function is:

$$r(s,a,s') \equiv \mathbb{E}\{R_{t+1}|S_t = s, A_t = a, S_{t+1} = s'\} = \frac{\sum_{r \in R} r p(s', r|s, a)}{p(s'|s,a)}$$

The best way to win is timid-timid-timid-timid-timid-bold.

In [42]:

```
import random
import numpy as np
import pandas as pd

def matchgame(way):
    if way ==1:
        random.seed(0)
        p = np.array([0.1, 0.9, 0.0])
        index = np.random.choice([-1, 0, 1], p = p.ravel())
    elif way ==2:
        random.seed(0)
        p = np.array([0.55, 0.0, 0.45])
        index = np.random.choice([-1, 0, 1], p = p.ravel())
    return index
i=0
result=[]
while i<10000:
    reward=0
    t=0
    while t<100:
        x=random.randint(1,2)
        if t<5:
            reward = reward+matchgame(x)
            t=t+1
        elif t>=5 and reward<1:
            reward = reward+matchgame(x)
            t=t+1
        elif t>=5 and matchgame(x)==1:
            result.append(t)
            break
    i=i+1

pd.value_counts(result).head(5)
```

Out[42]:

```
5      4130
7       654
9       412
11      290
13      220
dtype: int64
```

As a result, The most likely way to win is to play 5+1= 6 games. So the best way to win is timid-timid-timid-timid-timid-bold

Problem 2.

We have the Time Epochs $t = 1, 2 \ldots, 12$ (finite-horizon problems). Let A be the state of this software manufacturer, the state space $S = \{1,2\}$, the action space include two choices $A = \{0,1\}$, 0 means no investment in new development occurs, and 1 means the company invest in development of upgraded version of the software. When t=12, We should let the sum of all reward to be most.

The transition probability is:

$$p(s'|s,a) \equiv \Pr(S_{t+1} = s'|S_t = s, A_t = a) = \sum_{r \in R} p(s', r|s, a)$$

And the reward function is:

$$r(s,a,s') \equiv \mathbb{E}\{R_{t+1}|S_t = s, A_t = a, S_{t+1} = s'\} = \frac{\sum_{r \in R} rp(s', r|s, a)}{p(s'|s, a)}$$

If the initial state is state1, the explanation of best choice is 36.02778, and if the initial state is state 2, the explanation of best choice is 29.36111.

In [42]:

```
import sys
sys.setrecursionlimit(1000000000)
```

In [65]:

```
def sumreward(state,t,reward):
    while t<=12:
        if t==12:
            return 0
            break
        elif state==1:
            t=t+1
            a=6+0.5*sumreward(2,t,reward)+0.5*sumreward(1,t,rewa
rd)
            b=4+0.2*sumreward(2,t,reward)+0.8*sumreward(1,t,rewa
rd)
            if a>=b:
                reward=reward+a
```

```
                choice='0 '
            else:
                reward=reward+a
                choice='1 '
            return reward
        elif state==2:
            t=t+1
            c=1+sumreward(2,t,reward)
            d=-2+0.7*sumreward(1,t,reward)+0.3*sumreward(2,t,rew
ard)
            if c>=d:
                reward=reward+c
                choice='1 '
            else:
                reward=reward+d
                choice='0 '
            return reward

reward = 0
t=0
print(sumreward(1,t,reward))
print(sumreward(2,t,reward))
```

```
36.027777791999995
29.361111091199994
```

Problem 3.

1) We have the Time Epochs $t = 1, 2 \dots, T$ (infinite-horizon problems). Let A be the state of this equipment, the state space $S \sim P_j, P_j = \frac{\lambda^j}{j!} e^{-\lambda}, j = 0, 1, \cdots$, the action space include two choices $A = \{0,1\}$, 0 means continue, and 1 means replace.

The transition probability is:

$p(s'|s,a) \equiv \Pr(S_{t+1} = s'|S_t = s, A_t = a) = \sum_{r \in R} p(s', r|s,a)$

And the reward function is:

$$r(s,a,s') \equiv \mathbb{E}\{R_{t+1}|S_t = s, A_t = a, S_{t+1} = s'\} = \frac{\sum_{r \in R} r p(s', r|s,a)}{p(s'|s,a)}$$

Our objective may be to maximize the the difference between cost, salvage and revenue per period.

2)

In [6]:

```
import numpy as np
from typing import Sequence, Tuple
from scipy.stats import poisson
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
from matplotlib import cm
```

```
T: int = 20  # time steps
M: int = 20  # initial inventory
# the following are (price, poisson mean) pairs, i.e., elasticit
y
el: Sequence[Tuple[float, float]] = [
    (10.0, 0.3), (9.0, 0.8), (8.0, 1.6),
    (7.0, 2.7), (6.0, 4.1), (5.0, 7.2)
]

# v represents the Optimal Value Function (time, Inventory) -> E
[Sum of Sales Revenue]
v: np.ndarray = np.zeros((T + 1, M + 1))
pi: np.ndarray = np.zeros((T, M + 1))
rvs: Sequence = [poisson(l) for _, l in el]

for t in range(T - 1, -1, -1):
    for s in range(M + 1):
        q_vals = [sum(rvs[i].pmf(d) * (d * p + v[t + 1, s - d])
                      for d in range(s)) +
                  (1. - rvs[i].cdf(s - 1)) * s * p
                  for i, (p, _) in enumerate(el)]
        v[t, s] = np.max(q_vals)
        pi[t, s] = el[int(np.argmax(q_vals))][0]


x, y = np.meshgrid(range(M + 1), range(T))
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(x, y, pi, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```