Question 1

Part (a)

Suppose that i time periods have passed and the remaining budget is x. Then the next candidate comes in. We hire her according to some policy $\delta = (\delta_1, \delta_2, \delta_3, \delta_4)$. If the policy suggests we should make the hire, we get a reward to k with probability $p_1\delta_1 + p_2\delta_2 + p_3\delta_3 + p_4\delta_4$. and we expect to obtain some value in the remaining iterations, with the remaining budget, namely $V_{i+1}(x-1)$. Now if the policy suggests that we should not make the hire, we move to the next state, namely $V_{i+1}(x)$ with probability $1 - p_1\delta_1 - p_2\delta_2 - p_3\delta_3 - p_4\delta_4$. Therefore in order to determine the optimal policy at each state in order to maximize our value we just have to evaluate which of the two options gives us the highest value. Therefore, we have that

$$V_i(x) = \sum_{k=1}^{4} p_k \max\{k + V_{i+1}(x-1), V_{i+1}(x)\}$$

Note also that δ_k is in essence hidden in this equation. If the maximum function picks the left argument, then $\delta_k = 1$ for this particular k and $\delta_k = 0$ otherwise.

Part (b)

In [10]:
```
def rec(x, i, cache, horizon):

    if i > horizon:
        return 0
    if x == 0:
        return 0

    if (x,i) in cache:
        return(cache[(x,i)])

    s = 0
    for k in range(1, 5):
        s += 0.25 * max(k + rec(x - 1, i + 1, cache, horizon),
                        rec(x, i + 1, cache, horizon))

    cache[(x, i)] = s

    return s
```

We now compute compute $V_1(B)$ for different values of the budget B.

In [11]:
```
v = []

for i in range(10,110,10):
    v.append(rec(x = i , i = 1, cache = {}, horizon = 100))

v
```

Out[11]:
```
[39.9999052915732,
 79.66808259774882,
 113.88312226804393,
 143.98469879134097,
 172.1945363788912,
 193.98469879134103,
 213.88312226804396,
 229.66808259774882,
 239.99990529157316,
 250.0]
```

As a sanity check, notice that when we have $B = 100$, i.e we can hire someone at every step, then we get the average of the rankings at each step.

We now slightly modify the above function in order to print the optimal policy at each step.

In [12]:
```
def rec_policy(x, i, cache, horizon):

    if i > horizon:
        return 0
    if x == 0:
        return 0

    if (x,i) in cache:
        return(cache[(x,i)])

    s = 0
    found = False
    for k in range(1, 5):
        s += 0.25 * max(k + rec_policy(x - 1, i + 1, cache, horizon),
                        rec_policy(x, i + 1, cache, horizon))
        if found == False and (k + rec_policy(x - 1,i + 1,cache,horizon))\
            > rec_policy(x, i + 1, cache, horizon):
            print("Remaining To Spend: " + str(x)
                  + " Iteration: " + str(i) + " Hire: " + str(k) )
            found = True

    cache[(x, i)] = s

    return s
```

In [13]:
```
rec_policy(x = 5 , i = 80, cache = {}, horizon = 100)
```

```
Remaining To Spend: 1 Iteration: 100 Hire: 1
Remaining To Spend: 1 Iteration: 99 Hire: 3
Remaining To Spend: 1 Iteration: 98 Hire: 4
Remaining To Spend: 1 Iteration: 97 Hire: 4
Remaining To Spend: 1 Iteration: 96 Hire: 4
Remaining To Spend: 1 Iteration: 95 Hire: 4
Remaining To Spend: 1 Iteration: 94 Hire: 4
Remaining To Spend: 1 Iteration: 93 Hire: 4
Remaining To Spend: 1 Iteration: 92 Hire: 4
Remaining To Spend: 1 Iteration: 91 Hire: 4
Remaining To Spend: 1 Iteration: 90 Hire: 4
Remaining To Spend: 1 Iteration: 89 Hire: 4
Remaining To Spend: 1 Iteration: 88 Hire: 4
Remaining To Spend: 1 Iteration: 87 Hire: 4
Remaining To Spend: 1 Iteration: 86 Hire: 4
Remaining To Spend: 1 Iteration: 85 Hire: 4
Remaining To Spend: 1 Iteration: 84 Hire: 4
Remaining To Spend: 2 Iteration: 100 Hire: 1
Remaining To Spend: 2 Iteration: 99 Hire: 1
Remaining To Spend: 2 Iteration: 98 Hire: 3
Remaining To Spend: 2 Iteration: 97 Hire: 3
Remaining To Spend: 2 Iteration: 96 Hire: 3
Remaining To Spend: 2 Iteration: 95 Hire: 4
Remaining To Spend: 2 Iteration: 94 Hire: 4
Remaining To Spend: 2 Iteration: 93 Hire: 4
Remaining To Spend: 2 Iteration: 92 Hire: 4
Remaining To Spend: 2 Iteration: 91 Hire: 4
Remaining To Spend: 2 Iteration: 90 Hire: 4
Remaining To Spend: 2 Iteration: 89 Hire: 4
Remaining To Spend: 2 Iteration: 88 Hire: 4
Remaining To Spend: 2 Iteration: 87 Hire: 4
Remaining To Spend: 2 Iteration: 86 Hire: 4
Remaining To Spend: 2 Iteration: 85 Hire: 4
Remaining To Spend: 2 Iteration: 84 Hire: 4
Remaining To Spend: 2 Iteration: 83 Hire: 4
Remaining To Spend: 3 Iteration: 100 Hire: 1
Remaining To Spend: 3 Iteration: 99 Hire: 1
Remaining To Spend: 3 Iteration: 98 Hire: 1
Remaining To Spend: 3 Iteration: 97 Hire: 2
Remaining To Spend: 3 Iteration: 96 Hire: 3
Remaining To Spend: 3 Iteration: 95 Hire: 3
Remaining To Spend: 3 Iteration: 94 Hire: 3
Remaining To Spend: 3 Iteration: 93 Hire: 3
Remaining To Spend: 3 Iteration: 92 Hire: 4
Remaining To Spend: 3 Iteration: 91 Hire: 4
Remaining To Spend: 3 Iteration: 90 Hire: 4
Remaining To Spend: 3 Iteration: 89 Hire: 4
Remaining To Spend: 3 Iteration: 88 Hire: 4
Remaining To Spend: 3 Iteration: 87 Hire: 4
Remaining To Spend: 3 Iteration: 86 Hire: 4
Remaining To Spend: 3 Iteration: 85 Hire: 4
Remaining To Spend: 3 Iteration: 84 Hire: 4
Remaining To Spend: 3 Iteration: 83 Hire: 4
Remaining To Spend: 3 Iteration: 82 Hire: 4
Remaining To Spend: 4 Iteration: 100 Hire: 1
```

```
Remaining To Spend: 4 Iteration: 99 Hire: 1
Remaining To Spend: 4 Iteration: 98 Hire: 1
Remaining To Spend: 4 Iteration: 97 Hire: 1
Remaining To Spend: 4 Iteration: 96 Hire: 2
Remaining To Spend: 4 Iteration: 95 Hire: 2
Remaining To Spend: 4 Iteration: 94 Hire: 3
Remaining To Spend: 4 Iteration: 93 Hire: 3
Remaining To Spend: 4 Iteration: 92 Hire: 3
Remaining To Spend: 4 Iteration: 91 Hire: 3
Remaining To Spend: 4 Iteration: 90 Hire: 3
Remaining To Spend: 4 Iteration: 89 Hire: 4
Remaining To Spend: 4 Iteration: 88 Hire: 4
Remaining To Spend: 4 Iteration: 87 Hire: 4
Remaining To Spend: 4 Iteration: 86 Hire: 4
Remaining To Spend: 4 Iteration: 85 Hire: 4
Remaining To Spend: 4 Iteration: 84 Hire: 4
Remaining To Spend: 4 Iteration: 83 Hire: 4
Remaining To Spend: 4 Iteration: 82 Hire: 4
Remaining To Spend: 4 Iteration: 81 Hire: 4
Remaining To Spend: 5 Iteration: 100 Hire: 1
Remaining To Spend: 5 Iteration: 99 Hire: 1
Remaining To Spend: 5 Iteration: 98 Hire: 1
Remaining To Spend: 5 Iteration: 97 Hire: 1
Remaining To Spend: 5 Iteration: 96 Hire: 1
Remaining To Spend: 5 Iteration: 95 Hire: 2
Remaining To Spend: 5 Iteration: 94 Hire: 2
Remaining To Spend: 5 Iteration: 93 Hire: 3
Remaining To Spend: 5 Iteration: 92 Hire: 3
Remaining To Spend: 5 Iteration: 91 Hire: 3
Remaining To Spend: 5 Iteration: 90 Hire: 3
Remaining To Spend: 5 Iteration: 89 Hire: 3
Remaining To Spend: 5 Iteration: 88 Hire: 3
Remaining To Spend: 5 Iteration: 87 Hire: 3
Remaining To Spend: 5 Iteration: 86 Hire: 4
Remaining To Spend: 5 Iteration: 85 Hire: 4
Remaining To Spend: 5 Iteration: 84 Hire: 4
Remaining To Spend: 5 Iteration: 83 Hire: 4
Remaining To Spend: 5 Iteration: 82 Hire: 4
Remaining To Spend: 5 Iteration: 81 Hire: 4
Remaining To Spend: 5 Iteration: 80 Hire: 4
```

Out[13]: 19.022788893722463

We now have the value we are expecting to get under these conditions and also the hiring decision at each step. For example when we we have 1 more to spend and two people to interview, we should hire someone that is a 3 or more but when we only have 1 more people to interview we should hire anyone.

Part (c)

By inspecting the result obtained from part (b), we can see that this provides a good approximation to the actual value function, which is 113.9.

Question 2

Part (a)

We can use the equation derived above. However, we don't have the time index anymore since the horizon is infinite now. Furthermore, since a discounting of β is applied at every step, we must multiply the the value function with β in the formula. We thus have

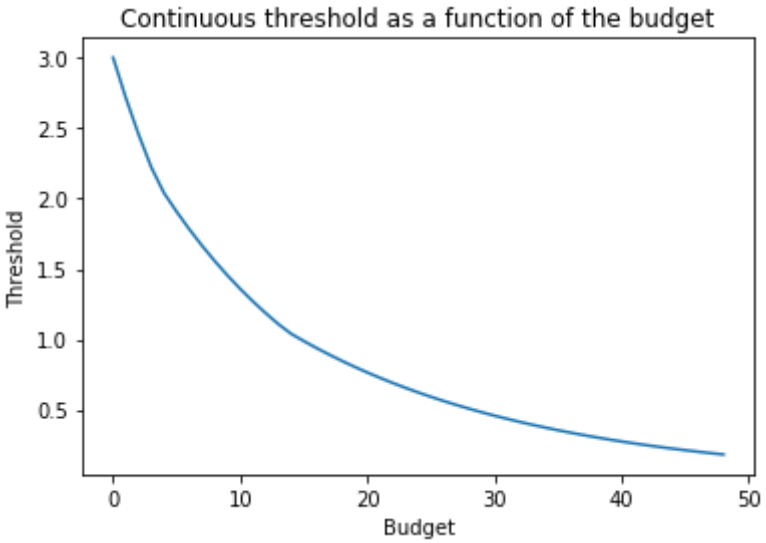
$$V(x) = \sum_{k=1}^{4} p_k \max\left\{k + \beta V(x-1), \beta V(x)\right\}$$

In [14]:
```
import numpy as np

beta = 0.95

v = np.zeros(51)
t = np.zeros(51)

for i in range(1000):
    for x in range(1,51):
        s = 0
        t_i = []
        for k in range(1,5):
            s += 0.25*max(k + beta*v[x-1], beta*v[x])
            if (k + beta*v[x-1]) > beta*v[x]:
                t_i.append(k)

        v[x] = s
        t[x] = min(t_i)

v
```

Out[14]:
```
array([ 0.        ,  3.47826087,  6.48033126,  9.19649019, 11.65396732,
       13.877399  , 15.91822529, 17.82522691, 19.60717925, 21.27228225,
       22.82819817, 24.28208681, 25.6406385 , 26.91010482, 28.09632746,
       29.204765  , 30.24452675, 31.23230041, 32.17068539, 33.06215112,
       33.90904357, 34.71359139, 35.47791182, 36.20401623, 36.89381542,
       37.54912465, 38.17166841, 38.76308499, 39.32493074, 39.85868421,
       40.36575   , 40.8474625 , 41.30508937, 41.7398349 , 42.15284316,
       42.545201  , 42.91794095, 43.2720439 , 43.60844171, 43.92801962,
       44.23161864, 44.52003771, 44.79403582, 45.05433403, 45.30161733,
       45.53653646, 45.75970964, 45.97172416, 46.17313795, 46.36448105,
       46.546257  ])
```

Part (b)

In [15]:
```
import matplotlib.pyplot as plt

threshold_cts = [v[i+1]-v[i] for i in range(1,len(t)-1)]
plt.plot(threshold_cts)
plt.title('Continuous threshold as a function of the budget')
plt.xlabel('Budget')
plt.ylabel('Threshold')
plt.show()
```



We can also plot a discrete version of this using the 't' list computed above. Here, for each value of budget we have a discrete value saying what is the minimum quality that an employee should have in order to get hired for each value of the budget.

In [16]:
```
plt.scatter(np.arange(51),t)
plt.title('Discrete threshold as a function of the budget')
plt.xlabel('Budget')
plt.ylabel('Threshold')
plt.show()
```