Question 2

The probability of not failing is $P(\text{success}) = \prod_{i=1}^{N}(1 - p_i^{n_i})$, where p_i is the probability of failure for the i-th component type and n_i is the number of selected components of the i-th type (say there are N components). We want to maximise $P(\text{success})$, or equivalently, minimise (minimise a sum to be consistent with the dynamic programming formulation in the lecture notes)

$$V = -\log(P(\text{success})) = \sum_{i=1}^{N} -\log(1 - p_i^{n_i})$$

subject to the budget constraint (where c_i are the costs):

$$\sum_{i=1}^{N} n_i c_i \le B$$

This is distinct from the unbounded knapsack problem in that the value of a component ($-\log(1 - p_i^{n_i})$) is non-linear in how many we choose (n_i). In the unbounded knapsack problem, the value obtained from selecting n of an item with value v is just nv, which makes for a simpler dynamic programming solution where the values can just be summed. Consequently, we need to treat the choice of n_i for one component type as a single decision, as otherwise we can't sum over values.

For our value function it's tempting to write (with B as the budget, and c_i as the costs):

$$V(B) = \min_{n_1,\dots,n_N} \left\{ \sum_{i=1}^{N} -\log(1 - p_i^{n_i}) \right\}$$

$$V(B) = \min_{n_1,\dots,n_N} \left\{ -\log(1 - p_1^{n_1}) + \sum_{i=2}^{N} -\log(1 - p_i^{n_i}) \right\}$$

$$V(B) = \min_{n_1} \{ -\log(1 - p_1^{n_1}) + V(B - n_1 c_1) \}$$

but this is wrong, since $V(B - n_i c_i)$ is the cost-to-go of the exact same problem, just with a smaller budget, implying that we'd be able to choose n_1 again in this subproblem (which we can't, for reasons described above).

Instead, we can add a state variable tracking which component types we have left to choose (similar to the solution of the 0/1 knapsack problem). The value function $V(k, B)$ represents the cost of the best solution to the problem with budget B and component types $1, 2, \dots, k$. So:

$V(0, B) = 0$ (a system with no components can never fail)

$V(i, 0) = \infty$ for all $i > 0$ (if a system has components, and we can't afford any, it will always fail)

$V(i, B) = \min_{n_i} \{ -\log(1 - p_i^{n_i}) + V(i - 1, B - n_i c_i) \}$ (Bellman equation)

So we can compute $V(i,b)$ for all $(i,b) \in \{1,2,...,N\} \times \{1,2,...,B\}$ and then backtrack for the solution. Each decision is the choice $n_i \in \left\{1,2,...,\left\lfloor \frac{b}{c_i} \right\rfloor\right\}$ that minimises $V(i,b)$.

```
budget = 2000;
costs = [100 200 100 200 300];
fail_probs = [0.2 0.1 0.3 0.15 0.05];

[p_success, components] = selectcomps(budget, costs, fail_probs)
```

```
p_success = 0.9317
components =
     3     2     3     2     2
```