Question 1

a Let $c(x,t,a)$ be the cost of starting in state x in time t and taking action a.

$$\begin{aligned}
c(0,t,0) &= 0\\
c(0,t,1) &= U + p(t) * 2\\
c(0,t,2) &= U + p(t) * 5\\
\\
c(1,t,0) &= D\\
c(1,t,1) &= p(t) * 2\\
c(1,t,2) &= p(t) * 5\\
\\
c(2,t,0) &= D\\
c(2,t,1) &= p(t) * 2\\
c(2,t,2) &= p(t) * 5
\end{aligned}$$

b Stages: $n \in (t,r)$ where $t = 1,2,\dots,12$ is the time period in hours and $r = 1000, 2000, \dots, 10000$ is the number of remaining doughuts.
States: $x \in 0,1,2$ corresponding to the mode of the machine.
Actions: $a \in 0,1,2$ corresponding to switching to the specified mode.
Rewards: Let us define $f(a)$ such that $f(a) = \{5 \text{ if } a = 2, 2 \text{ if } a = 1, 0 \text{ if } a = 0\}$. The reward function is then:

$$c_n(x,a) = \begin{cases} U + f(a) * p(n) & \text{when } x = 0, a \in \{1,2\} \\ f(a) * p(n) & \text{when } x \in \{1,2\}, a \in \{1,2\} \\ D & \text{when } x \in \{1,2\}, a = 0 \\ 0 & \text{when } x = 0, a = 0 \end{cases}$$

c

$$V_{13}(\cdot) = \begin{cases} 0 & \text{when } r \geq 10000, x = 0 \\ D & \text{when } r \geq 10000, x \in \{1,2\} \\ \infty & \text{otherwise} \end{cases}$$

d

$$V_n(x,r) = \min_{a \in A} \left\{ c_n(x,a) + \sum_{X_{n+1}} \rho_n(x,y,a) * V_{n+1}(y, r - 1000 * a) \right\}$$

where:

$$\rho_n(x,y,a) = \begin{cases} 1 & \text{when } y = a \\ 0 & \text{otherwise} \end{cases}$$

which expands to:

$$V_n(x,r) = \min \left\{ c_n(x,0) + V_{n+1}(0,r),\ c_n(x,1) + V_{n+1}(1, r - 1000),\ c_n(x,2) + V_{n+1}(2, r - 2000) \right\}$$

e dynamic.m

```
%This script runs the dynamic programming recursion and determines the
%optimal actions to take in each period.
%
% V is a vector of the total cost of stages t,t+1,...,12, if you are
    ↪ in
% state (x,r) in stage t. (Use the getIndex function to modify it.)

% actions is a vector of the optimal actions, given the state (x,r)
    ↪ and the
% stage t.

% M is a big-M used to set the value of infeasible states in the final
% stage, and also to initialize unexplored states.

% U is the start-up cost, D is the shut-down cost, and p is a vector
    ↪ of
% electricity prices.

clear all;
M=10000;

U=10;
D=10;
p=[40;42;38;29;25;26;38;16;24;32;36;48];

V=M*ones(3*11*13,1);

V(getIndex(0,0,13))=0;
V(getIndex(1,0,13))=D;
V(getIndex(2,0,13))=D;

actions=zeros(3*11*13,1);

for t = 12:-1:1
    for x=0:2
        for r=0:1000:10000
            index=getIndex(x,r,t);
            for a=[0, min(r/1000, 1), min(r/1000, 2)]
                next_index=getIndex(a,r-1000*a,t+1);
                temp=stage_cost(x,a,t,U,D,p) + V(next_index); %finds
                    ↪ the stage cost + future cost of action a
                if temp<V(index) %if this action is cheaper than other
                    ↪  actions, so far
                    V(index)=temp; %set the value of this action to be
                        ↪  the value of being in this state/stage
                    actions(index)=a; %set the current action to be
                        ↪ the best action
                end
            end
        end
    end
end
```

stage_cost.m

```
function cost = stage_cost(x, a, t, U, D, p)
%This function takes in the state of the machine at the beginning of
    ↪ hour t
%and the action that is taken in hour t and returns the stage-cost.
%
%Inputs:    x        the state of the machine (0,1,2) at beginning of
    ↪ hour t
%           a        the action we choose at start of hour t
%           t        the period (hour)
%           U        the start-up cost
%           D        the shut-down cost
%           p        the vector of prices over the 12 periods

    if(x==0)
        if(a==0)
            cost=0;
        else
            cost=U + floor(2.5*a)*p(t);
        end
    else
        if(a==0)
            cost=D;
        else
            cost=floor(2.5*a)*p(t);
        end
    end
end
```

f Using U = D = 10:
Optimal policy = 000111122110 ie enter mode 1 in time 4, enter mode 2 in time 8, enter mode 1 in time 10, shut down in time 12. This returns a final cost of \$592.

Using U = D = 0:
Optimal policy = 000121022110 where entering modes follows the logic shown above. This returns a final cost of \$571.

When there are no start-up and shut-down costs, the final cost is lower by \$21. This is due to the machine being able to take advantage of lower cost time periods without needing to worry about the cost of switching modes. We can see this in the extra start-up and shut-down which would normally be penalised by U and D.

g Redefine ρ as:

$$\rho_n(x,y,a) = \begin{cases} 0.25 & \text{when } x=0, y=0, a\in\{1,2\} \\ 0.75 & \text{when } x=0, y=a\in\{1,2\} \\ 1 & \text{when } x=1, y=a \\ 0 & \text{otherwise} \end{cases}$$

Then for $x \in \{1,2\}$, it is unchanged:

$$V_n(x,r) = \min\left\{c_n(x,0)+V_{n+1}(0,r),\ c_n(x,1)+V_{n+1}(1,r-1000),\ c_n(x,2)+V_{n+1}(2,r-2000)\right\}$$

For $x = 0$:

$$\begin{aligned} V_n(0,r) = \min\Big\{ & c_n(0,0)+V_{n+1}(0,r), \\ & 0.75*(c_n(0,1)+V_{n+1}(1,r-1000))+0.25*(c_n(0,0)+V_{n+1}(0,r)), \\ & 0.75*(c_n(0,2)+V_{n+1}(2,r-2000))+0.25*(c_n(0,0)+V_{n+1}(0,r))\Big\} \end{aligned}$$

h dynamic.m

```
%This script runs the dynamic programming recursion and determines the
%optimal actions to take in each period.
%
% V is a vector of the total cost of stages t,t+1,...,12, if you are
    ↪ in
% state (x,r) in stage t. (Use the getIndex function to modify it.)

% actions is a vector of the optimal actions, given the state (x,r)
    ↪ and the
% stage t.

% M is a big-M used to set the value of infeasible states in the final
% stage, and also to initialize unexplored states.

% U is the start-up cost, D is the shut-down cost, and p is a vector
    ↪ of
% electricity prices.

clear all;
M=1e6;

U=0;
D=0;
p=[40;42;38;29;25;26;38;16;24;32;36;48];

V=M*ones(3*11*13,1);

V(getIndex(0,0,13))=0;
V(getIndex(1,0,13))=D;
V(getIndex(2,0,13))=D;

actions=zeros(3*11*13,1);

for t = 12:-1:1
    for x=0:2
        for r=0:1000:10000
            index=getIndex(x,r,t);
            for a=[0, min(r/1000, 1), min(r/1000, 2)]
                next_index_fail=getIndex(0,r,t+1);
                next_index=getIndex(a,r-1000*a,t+1);
                if (x==0 && a ~=0)
                    temp=0.75*(stage_cost(x,a,t,U,D,p) + V(next_index)
                        ↪ ) + 0.25*(stage_cost(x,0,t,U,D,p) + V(
                        ↪ next_index_fail));
                else
                    temp=stage_cost(x,a,t,U,D,p) + V(next_index); %
                        ↪ finds the stage cost + future cost of action
                        ↪  a
                end
                if temp<V(index) %if this action is cheaper than other
                    ↪  actions, so far
                    V(index)=temp; %set the value of this action to be
                        ↪  the value of being in this state/stage
                    actions(index)=a; %set the current action to be
                        ↪ the best action
                end
            end
        end
    end
end
```

i Policy = 111111121000, so we attempt to start up the machine as soon as possible, to account for potential failures.

Question 2

a

$$\begin{aligned} V_N &= \max\{\mathrm{E}[r], 0\} \\ &= \mathrm{E}[r] \\ &= \int_0^3 r * f(r)\,\mathrm{d}r \\ &= \int_0^3 r * \left(\frac{6-2r}{9}\right)\mathrm{d}r \\ &= \frac{1}{9}\left[3r^2 - \frac{2}{3}r^3\right]_0^3 \\ &= 1 \end{aligned}$$

b Accept if $R \geq 1$
Reject otherwise

c Let stages: $n = 1, 2, \ldots, N$
States: $x = (h, R)$ where $h = 1$ if we have accepted a candidate, 0 otherwise, and R being the reward from the current candidate.

Our recursion is then:

$$V_n(h, R) = \max\{R, \mathcal{V}_{n+1}\}$$

Integrate over r:

$$\int_0^3 f(r) * V_n(h, r)\,\mathrm{d}r = \int_0^3 f(r) * \max\{r, \mathcal{V}_{n+1}\}\,\mathrm{d}r$$

Recognise that the LHS is equal to $\mathcal{V}_n$ and the max function can be split into two integrals:

$$\begin{aligned} \mathcal{V}_n &= \int_0^{\mathcal{V}_{n+1}} f(r) * \mathcal{V}_{n+1}\,\mathrm{d}r + \int_{\mathcal{V}_{n+1}}^3 f(r) * r\,\mathrm{d}r \\ &= \frac{1}{9}\mathcal{V}_{n+1}\left[6r - r^2\right]_0^{\mathcal{V}_{n+1}} + \frac{1}{9}\left[3r^2 - \frac{2}{3}\right]_{\mathcal{V}_{n+1}}^3 \\ &= \frac{1}{9}\left(6\mathcal{V}_{n+1}^2 - \mathcal{V}_{n+1}^3\right) + 1 - \frac{1}{9}\left(3\mathcal{V}_{n+1}^2 - \frac{2}{3}\mathcal{V}_{n+1}^3\right) \\ &= 1 + \frac{1}{9}\left(3\mathcal{V}_{n+1}^2 - \frac{1}{3}\mathcal{V}_{n+1}^3\right) \\ &= 1 + \frac{1}{3}\mathcal{V}_{n+1}^2 - \frac{1}{27}\mathcal{V}_{n+1}^3 \end{aligned}$$

This gives values of 3 and 27 for x and y respectively.