Question 3:

The question only requires a "brief explanation" and not a "proof" so we can be a little less formal than on homework. After the inner for loop, $m = \prod_{j=1}^{i} j^2 = \left(\prod_{j=1}^{i} j\right)^2 = (i!)^2$. After the rth iteration of the inner while loop, $k = 2^{2^r}$. (If this were a "proof" I might do a quick induction to show this). So the inner while loop ends about when $2^{2^r} = (i!)^2$. Taking the log (base 2) of both sides we get $2^r = 2\log(i!) = 2\Theta(i\log(i))$. Taking the log again we get $r = \log(2\Theta(i\log(i))) = \log(2) + \Theta(\log(i) + \log(\log(i))) = \Theta(\log(i))$. So each iteration F is called $\Theta(\log(i))$ times. So in total F is called $\sum_{i=1}^{n}\Theta(\log(i))$ times. This is $O(n\log(n))$ since $\sum_{i=1}^{n}\Theta(\log(i)) \le \sum_{i=1}^{n}\Theta(\log(n)) = \Theta(n\log(n))$. This is $\Omega(n\log(n))$ since $\sum_{i=1}^{n}\Theta(\log(i)) \ge \sum i = n/2^n\Theta(\log(n/2)) = \Theta(n\log(n))$. Therefore F is called in total $\Theta(n\log(n))$ times.

Question 10:

(on scratch paper: $T(1) = 1, T(2) = 5, T(3) = 17, T(4) = 53$ compared to $3^1 = 3, 3^2 = 9, 3^3 = 27, 3^4 = 81$.)

We claim, and will show by induction, that $T(n) < \frac{2}{3}3^n$ for all $n \ge 1$.

Base case: At $n = 1$ we are given that $T(1) = 1 < 2$.

Inductive hypothesis: $T(n-1) < 3^{n-1}$ for some $n \ge 2$ Inductive step: By the recurrence, $T(n) = 3T(n-1) + 2 < 2 \cdot 3^{n-1} + 2 = \frac{2}{3}3^n + 2 \le 3^n - \frac{1}{3}3^2 + 2 < 3^n$.

Thus by induction, $T(n) \le \frac{2}{3}3^n$ for all $n \ge 1$. This shows that $T(n) = O(3^n)$ for $n \ge 1$.

Question 15:

1. Let $OPT(i)$ be the value of the optimal solution on elements $A[1]$ to $A[i]$. If $i < 2$, then we can't pick any pairs and $OPT(i) = 0$. Otherwise, we have a choice to make: we can either include $A[i-1], A[i]$ as a pair, or not. If we do, then we'll get $A[i-1] + A[i]$ value, but we won't be able to include $A[i-2]$ in any pairs. So our total value in this case will be $A[i-1] + A[i] + OPT(i-3)$. If we don't, then we can't include $A[i]$ in any pair, so our total value will be just $OPT(i-1)$. Taking these options together, we get a recurrence of $OPT(i) = \max(A[i-1] + A[i] + OPT(i-3), OPT(i-1))$.

2. Given the prior values of OPT, computing $OPT(i)$ takes $O(1)$ time; we're just adding and comparing a constant number of things. We need to compute $OPT(i)$ for $i = 2, 3, \ldots n$ to compute $OPT(n)$, so in total we need $O(n)$ time to compute $OPT(n)$.

3.

i	2	3	4	5
$OPT(i)$	8	8	11	22

The maximum value for array A is 22.