13.

a. This is a case of the Master theorem with $a = 5$, $b = 2$, $d = 1$.

As $a > b^d$, the running time is $O(n^{\log_b a}) = O(n^{\log_2 5}) = O(n^{2.33})$

b. $T(n) = 2T(n-1) + C$, for some constant C. $T(n)$ can then be expanded to

$$C\sum_{i=0}^{n-1} 2^i + 2^n T(0) = O(2^n)$$

c. This is a case of the Master theorem with $a = 9$, $b = 3$, $d = 2$.

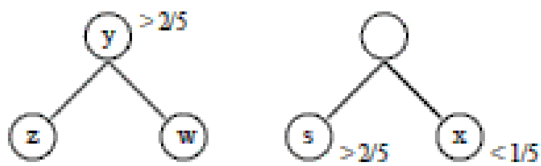
As $a = b^d$, the running time is $O(n^d \log n) = O(n^2 \log n)$

14.

a. Let s be the symbol with the highest frequency (probability) $p(s) > 2/5$ and suppose that it merges with some other symbol during the process of constructing the tree and hence does not correspond to a codeword of length 1. To be merged with some node, the node s and some other node x must be the two with minimum frequencies. This means there was at least one other node y (formed by merging of other nodes), with $p(y) > p(s)$ and $p(y) > p(x)$.

   Thus, $p(y) > 2/5$ and hence $p(x) < 1/5$.

   Now, y must have been formed by merging some two nodes z and w with at least one of them having probability greater than 1/5 (as they add up to more than 2/5). But this is a contradiction - $p(z)$ and $p(w)$ could not have been the minimum since $p(x) < 1/5$.



b. Suppose this is not the case. Let x be a node corresponding to a single character with $p(x) < 1/3$ such that the encoding of x is of length 1. Then x must not merge with any other node till the end. Consider the stage when there are only three leaves - x, y and z left in the tree. At the last stage y, z must merge to form another node so that x still corresponds to a codeword of length 1.

   But, $p(x) + p(y) + p(z) = 1$ and $p(x) < 1/3$ implies $p(y) + p(z) > 2/3$. Hence, at least one of $p(y)$ or $p(z)$, say $p(z)$, must be greater than $1/3$. But then these two cannot merge since $p(x)$ and $p(y)$ would be the minimum.

   This leads to a contradiction.

15.

The terms in the Fibonacci sequence are given by :

$F_1 = 1, \ F_2 = 1; \quad F_n = F_{n-1} + F_{n-2}$

(a) Give the pseudocode for a recursive algorithm to calculate the nth term in the Fibonacci sequence. Sample correct answer below.

```
fib(n)
        if n <= 2
           return  1
        else
          return fib(n-1)+fib(n-2)
```

(b) Describe in words and give the pseudocode for a dynamic programming algorithm to compute the nth term in the Fibonacci sequence.

Two versions memorized or Bottom Up. Only have to provide one.

Memo

```
memo = {}
fib (n) {
   if (n in memo) { return memo[n] }
   if (n <= 1) {
      f =  n;
    } else {
      f =  fib(n-1) + fib(n-2);
   }
   memo[n] = f;
   return f
}
```

Bottom-UP

```
fib = { }

   fib[1] = 1;

   fib[2] = 1;

   for k = 3 to n

        fib[k] =  fib[k-1] + fib[k-2];

   return fib[n]
```

(c) What is the running time of the DP algorithm.

DP is $\Theta(n)$

(d) How does this compare to the running time of the Recursive algorithm?

The recursive algorithm is **exponential $O(2^n)$ or $\Theta(\phi^n)$**. The DP is much faster since it is linear time.

16.

a)

$w_1 = 3, v_1 = 25$
$w_2 = 2, v_2 = 20$
$w_3 = 1, v_3 = 15$
$w_4 = 4, v_4 = 40$
$w_5 = 5, v_5 = 50$

i/j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	25	25	45	45
3	0	15	20	35	40	45	60
4	0	15	20	35	40	55	60
5	0	15	20	35	40	55	65

The maximal value of a feasible subset is F[5,6] = 65. The optimal subset is {item 3, item 5}.

b) The instance has a unique optimal subset

c) Time efficiency: $\Theta(nW)$
Space efficiency: $\Theta(nW)$
Time needed: $O(n)$