1.

a)

Iteration	Charlette	Elizabeth	Jane	Lydia	Purposal
1	Darcey				Charlette ⇒ Darcy
2	Darcey	Wicham			Elizabeth ⇒ Wicham
3	Darcey	Wicham	Colins		Jane ⇒ Colin
4	Darcey	Wicham	Colins		Lydia ⇒ Colin
5	Darcey	Wicham	Colins	Bingly	Lydia ⇒ Bingly

b)

Iteration	Charlette	Elizabeth	Jane	Lydia	Purposal
1	Jane				Bingly ⇒ Jane
2		Jane			Colin ⇒ Jane
3		Jane	Lydia		Darcey ⇒ Lydia
4		Jane	Lydia	Elizabeth	Wicham ⇒ Elizabeth
5	Charlette	Jane	Lydia	Elizabeth	Bingley ⇒ Charlette

c)

Base Case:
There are no pairing of men and women, m = 0 w = 0 S returns empty which is still stable
Induction Step:
Assume all pairs in S are stable. If an Engaged woman was previously engaged to another male, who that she prefers the previous male over her current fiancee.

Case 1:
Male is already paired but prefers new proposal
Assume that male paired with woman, w2, is in S which is stable. Woman, w1, sends a proposal to male, male is already engaged, by this case, which means another woman, w2, already sent him a proposal. Woman, w1, is higher on the males preferred list which means we will remove the pair male and w2. S still only contains stable pairs. Then the new pair, male and w1 is added to S. S contains only stable pairs. S still only contains stable pairs proving our assumption.
Case 2:
male is not paired
If male is not paired that means no woman has given him a proposal. Meaning no woman has that male as there preferred male on their list. New pair is added to S. S contains only stable pairs meaning our assumption is still true.
Case 3:
male is paired and is with preferred woman
Assume male is paired with woman, w2, is in S which is stable. If male prefers their current relationship then the woman, w1, who sent the proposal must go to the next best choice which means that that w1 prefers the man that she previously proposed to on the next iteration. S still only contains stable pairs meaning our assumption is still true.

d)

Same proof as c except change inductive step to assume S does not contain stable pairs. If we walk through the cases in c case 2 and 3 are still true, but case 1 is not. Case 1 will return an stable pair which is not our new assumption. Example:
Case 1:
Male is already paired but prefers new proposal
Assume that male paired with woman, w2 is in S which is NOT stable. Woman, w1, sends a proposal to male, male is already engaged, which, by definition, means that another woman, w2 already sent him a proposal. Woman, w1, is higher on the males preferred list which means we will remove the pair male and w2. S contains a stable pair. Means our assumption is not true, which is a contradiction.

2.

a)

Loop invariant:
After k loop iterations $y = \sum_{i=0}^{k} a_i * x^i$
Base Case:
$n = 0$ obvious
Inductive Step:
Assume $y = \sum_{i=0}^{k} a_i * x^i$ prior to the next loop $k + 1$
if $y_k = \sum_{i=0}^{k} a_i * x^i$, then $y_{k+1} = \sum_{i=0}^{k+1} a_i * x^i$
$y_{k+1} = \sum_{i=0}^{k+1} a_i * x^i$ can be written as $y_k + a_{k+1} * x^{k+1}$, obvious
this can be rewritten as $y_{k+1} = \sum_{i=0}^{k} a_i * x^i + a_{k+1} * x^{k+1}$, obvious
this can be rewritten as $y_{k+1} = \sum_{i=0}^{k+1} a_i * x^i$ by adding the summation and $a_{k+1} * x^{k+1}$
this is our assumption, proving it true

b)

Base Case:
$n = 0$, obvious
Inductive Step:
Assume k ¡ n, show n
Before while loop:
$y = a_n, i = n$
Therefore $y = a_i$
Inside loop:
$i = i - 1, y = a_i + y * x$
$a_i = a_{i-1}$ by what is done before the loop
$y = a_{i-1} + a_i * x$
Outside loop:
$y = ai - 1 + a_i * x$ by what is done from the loop
Don't know what to do after this "wave-hands" and *poof*
$\sum_{i=0}^{i} a_i * x^i = a_{i-1} + a_n * x$

3.

a)

function(n)
S = , empty set
if(n ==0)
return S
end if
S^1 = function(n-1)
if(S^1 ==), empty set
S^1.add(""), add blank string
else

foreach(x in S^1), start loop
S.add("1x")
S.add("0x")
end if
return S
end function

b)

Loop Invariant:
After k loop iterations $S = thesetofallcombinationofbinarystringlengthn$ which is 2^n
Base Case:
n = 0, obvious
Inductive Step: Strong Induction
Assume all k ¡ n are true, show true for n
if $n = 1, 2^n = |function(n-1)|+1$, —function(n-1)— means cardinality if $n > 12^n = |function(n-1)|*2$
if $n = 0$, base case obvious
Case 1:
n = 0
Base case, obvious
Case 2:
n = 1
Assume that function(n-1) returns true which means we will plug in function(0)
function(n-1) + 1, we add 1 because if n = 1 that means the recursive call, function(0) will return an empty set, base case. This means that we will go through the if statement which will add an empty string to the set, + 1.
We will then hit the foreach loop, and repeat for every element, which in this case is 1
this will return a set of cardinality 2 which can be written as 2^1. This is our assumption which is 2^n where n=1.
Case 3:
$n > 1$
$2^n = |function(n-1)| * 2$, —function(n-1)— is cardinality
We have the times because in the foreach loop we will walk through every element and return 2 values which is the same as the cardinality of —function(n-1) * 2
Assume function(n-1) will return the correct value, 2^{n-1} $2^{n-1} * 2$ can be rewritten as 2^n which is our assumption.

4.

a)

$p = 3$
$(3, 4, 2, 6, 9, 8, 10, 7)$

b)

mystery algorithm return the element at k when that element is the pivot point. Meaning all values to the left of it are less than element at k and all values to the right of it are greater or equal to element at k.

c)

Loop Invariant:
n = right-left + 1
Base Case:
n = 1, right = left, obvious
Induction Step:

Assume k ¡ m is true, show m
Case 1: k = p
Assume partition algorithm returns what it is suppose to p.
By definition of the partition algorithm $left \leq p \leq right$ and $right \leq n-1$.
That means biggest number p can be is n-1 which is less than m.
With our assumption p=k, that means k¡n
Case 2: $k < p$
Assume partition algorithm returns correct answer, p
same as case 1 except $p > k, p \leq n-1 then k < n-1 < n$
we get $k < n$ which is our assumption.
Case 3: $k > p$
Assume partition algorithm return correct answer, p
by definition of partition algorithm if case 1 or case 2 aren't true then by $p < k \leq right$
$k \leq right$ which means $k \leq n-1$
$n > n-1$ therefore $k < n$ which is our assumption