1. (8 points, 4 points each)

d,a,b,c This was my mistake, give points for this answer or to anyone who argues that the others are not subsets of c
d,a,b,c

2. (10 points, 2 points each) Take off a point if an explanation or counter example is not given for not valid answers.

a) Not valid, x could be anything, not necessarily 0
b) Valid
c) Valid
d) Valid
e) Valid

3. (8 points, 2 points each) Take off a point if an explanation, logical argument, or counter example is not given for not valid answers. Make sure the answer makes sense.

For example,
a: x > 0; b: x >= 0; c: x>= -1; d: x>= -2
code: x = x+1
x: x > 1; y: x > 0; z: x > -1

a) Not valid
{x >= -1} x = x + 1 {x > 0} – not valid x could initially be -1
b) Not valid; if x >= -2, x+1; x > 1 conclusion x > 1 doesn't hold if x == -2.
c) Valid
d) Valid

4. (6 points, 2 points each condition, 1 point off for not simplifying postcondition, They don't have to simplify intermediate steps.,

{ x > 0 }
y = x +1;
{x> 0 && y==x+1} = {x > 0 && y > 1}
z = y;
{x > 0 && y > 1 && z == y} = {x > 0 && y > 1 && z > 1}
z = z + 1;
{ z == z_old + 1 && x > 0 && y > 1 && z_old > 1} = {x > 0 && y > 1 && z > 2}

They don't have to carry x and y to final conclusion, but z must be simplified.
Watch out for equivalences like y > 1 == y >= 2 for ints.

5.

(18 points, 2 points each condition)

They don't have to write it as wp(..) = …

If you have doubts about any answers, ask me.

a) (10 pts) Take of a point if final wp is not simplified to something reasonable. They should have three conditions for y: y >= 0, y == -1, y < -1.

(y >= 0 && x == 4) || ((y < 0 && ((y == -1 && x == 3) || (y != -1 && (x == -2 || x == 2))) =

(y >= 0 && x = 4) || (y == -1 && x == 3) || (y < -1 && (x = -2 || x = 2))

```
if (y >= 0) {
        wp(z = x, z == 4) = (x == 4)
    z = x;
}
else {
    (y == -1 && x = 3) || (y != -1 && (x == -2 || x == 2))
    if( y == -1) {
            wp(z = x + 1, z == 4) = (x == 3)
        z = x + 1;
    }
    else {
            wp(z = x* x, z == 4) = (x == -2 || x == 2)// deduct a point if only one value is given
        z = x * x;
    }
}
{z == 4}
```

b) (4 points)

P: {wp(b=b+1, (b=-1) || (b=-2)) = (b=-2)||(b=-3)}

b = b+1

{wp(a = b+3, a>0 && b<0) = (b+3>0 && b <0) = ((b> -3) && (b < 0))= ((b=-1) || (b=-2)}

a = b + 3

{a > 0 && b < 0}

c) (4 points)

wp(y =4*w – 10, 2*x = y) = (x = 2*w – 5)

y = 4*w –10;

wp(x = 2*x, x == y) = (2*x = y)

x = 2 * x;

{ x == y }

Watch for equivalence line y < -1 == y <= -2 for ints.

6. (18 points, 2 points each)

a) true
b) false
c) false
d) true
e) false
f) false

g) true
h) true
i) true

7. (12 points)

a) (2 points)

D = arr.length – n
D(k) = arr.length – k
D(k+1) = arr.length – (k+1) = D(k) - 1 => D(k+1) < D(k)
(D==0) => (n == arr.length) loop exit condition. Could rewrite the loop as while(D>0) {…}
If they just argue that n increases until it reaches arr.length, give points.

b) (4 points)

LI: m >= arr[i-1] forall i :: 1 <= i <= n && n <= arr.length
Before loop m = arr[0]; n == 1. So m >= arr[0]
Take off a point if n <= arr.lengh is missing
The can express the loop invariant in text or something reasonable like
m = maximum(arr[0...n-1]) && n <= arr.length

c) (6 points)

n(k) is n at end of iteration k.
assume m(k) >= arr[i-1] forall i :: 1 <= i <= k && k <= arr.length
If arr[n(k+1)-1] > m(k), // we're comparing arr[n] to previous values of m before n(k) is updated
arr[n(k+1) -1] > m(k)
arr[n(k+1)-1] > arr[i-1] forall i :: 1 <= i <= k
arr[n(k+1)-1] >= arr[i-1] forall i :: 1 <= i <= k+1
m(k+1) >= arr[i-1] forall i :: 1 <= i <= k+1
else
since m hasn't changed
m(k+1) == m(k) >= arr[i-1] forall i :: 1 <= i <= k == arr[i-1] forall i :: 1 <= i <= k+1
m(k+1) >= arr[i-1] forall i :: 1 <= i <= k+1

n(k+1) = n(k) + 1
n(k) < arr.length or we would have exited loop so n(k+1) <= arr.length

Give points for reasonable arguments
They don't have to be so formal, but they must argue that either the m increases to arr[n-1] or stays the same.
Be careful, student notation may be hard to decipher. Ask if in doubt.

8. (10 points)
a) (5 points)

Requires: none (take off ½ point if they say element is not null, spec doesn't say that)

Modifies: this

Effects: this_post[index] = element,
this_post[i] forall i :: index <= i < size() -> this_post[i+1] = this_pre[i]

Returns: none

Throws: IndexOutOfBoundsException

They can describe the effects as text.

b) (5 points)

Give credit if the logic here matches the spec above, even if the spec contains errors. They don't have to be so formal describing the effects, a text description of element insertion is OK as long as the logic is correct. It's OK if they don't specify that nothing else is modified. That's implied by effects. Writing code is not a logical expression.

```
R = true
E = if Index < 0 || Index >= size())
        throw IndexOutOfBoundsException
    else
        this_post[index] = element,
        this_post[i] forall i :: index <= i < size() -> this_post[i+1] = this_pre[i]
true => E && (nothing else is modified)
```

Question 9. (10 points, 5 points each, ignore syntax errors)

a)

```
CharSet s = new CharSet()       // don't deduct points if they neglect declarations or print
Character x = new Character('x');
s.insert(x);
s.insert(x);
s.delete(x);
if(s.member(x))
        System.out.println("Error");
```

Take off a point if they directly access elts. elts is private
Take off a point if they don't use member functions
They could also check size() after inserting the same value twice.

b)

```
public void insert(Character c) {
        if (member(c))
                return;
        elts.add(c);
}
```

Take off a point if they return a value from insert