Here are the solutions in Dafny:

http://rise4fun.com/Dafny/9jRE - Complete Q1
http://rise4fun.com/Dafny/ggsn - Complete Q2
http://rise4fun.com/Dafny/vZjw7 - Complete Q3

Q1

c) @pre, @post, @L0, @L1 are respectively precondition, postcondition, first loop invariant, second loop invariant.

(1):
```
@pre
i := 1
@L0
```

(2):
```
@L0
assume i>=a.Length
@post
```

(3):
```
@L0
assume i<a.Length
j:=i
value:= a[i]
a[i]:=a[i-1]
@L1
```

(4):
```
@L1
assume j>0 && a[j-1]>value
a[j] := a[j-1]
j := j-1
@L1
```

(5):
```
@L1
assume j=<0 || a[j-1]=<value
a[j]:=value
i:=i+1
@L0
```

d)

```
@L1:sorted(a,0,i+1) && (i==j || Min(a,j,i) >= value)
S0: assume j>0 && a[j-1]>value
S1: a[j] := a[j-1]
S2: j := j-1
@L1:sorted(a,0,i+1) && (i==j || Min(a,j,i) >= value)
```

```
  @L1 ⇒ wp(@L1, S0;S1;S2)
= @L1 ⇒ wp(wp(@L1, j:=j-1), S0;S1)
= @L1 ⇒ wp(@L1{j ↦ j-1}, S0;S1)
= @L1 ⇒ wp(wp(@L1{j ↦ j-1},a[j]:=a[j-1]), S0)
= @L1 ⇒ wp(@L1{j ↦ j-1, a[j] ↦ a[j-1]}, S0)
= @L1 ⇒ wp(@L1{j ↦ j-1, a[j] ↦ a[j-1]}, j>0 && a[j-1]>value)
= @L1 ⇒  (j>0 && a[j-1]>value ⇒ @L1{j ↦ j-1, a[j] ↦ a[j-1]})
```

Q3

b)

From the program,

```
@pre: a != null && 0 <= left <= right < a.Length;
@post: sorted(a,left,right) &&
      forall i::(0<=i<left || right<i<a.Length) ==> a[i] == old(a[i])
      a[left] == min(a,left,right);
      a[right] == max(a,left,right);
      min(a, left, right) == old(min(a, left, right));
      max(a, left, right) == old(max(a, left, right));
```

```
@pre:
S0: assume a[left]>a[right];
S1: tmp:=a[left];
S2: a[left]:=a[right];
S3: a[right]:=tmp;
S4: assume left+1<right;
S5: k:=(right-left+1)/3;
@post:
```

```
  @pre ⇒ wp(@post, S0;S1;S2;S3;S4;S5)
= @pre ⇒ wp(@post{k ↦ (right-left+1)/3}, S0;S1;S2;S3;S4)
= @pre ⇒ wp((left+1<right) ⇒ @post{k ↦ (right-left+1)/3}, S0;S1;S2;S3)
= @pre ⇒ wp((left+1<right) ⇒ @post{k ↦ (right-left+1)/3, a[right] ↦ tmp}, S0;S1;S2)
= @pre ⇒ wp((left+1<right) ⇒ @post{k ↦ (right-left+1)/3, a[right] ↦ tmp, a[left] ↦ a[right]}, S0;S1)
= @pre ⇒ wp((left+1<right) ⇒ @post{k ↦ (right-left+1)/3, a[right] ↦ a[left], a[left] ↦ a[right], tmp ↦
a[left]}, S0)
= @pre ⇒ a[left]>a[right] ⇒ ((left+1<right) ⇒ @post{k ↦ (right-left+1)/3, a[right] ↦ a[left], a[left] ↦
a[right], tmp ↦ a[left]})
= @pre ⇒ (a[left]>a[right] && left+1<right) ⇒ @post{k ↦ (right-left+1)/3, a[right] ↦ a[left], a[left] ↦
a[right], tmp ↦ a[left]}
```


Q4

a)

@pre
assume l ≤ u
m := (l + u) div 2
assume a[m] = e
@post

@pre
assume l ≤ u
m := (l + u) div 2
assume a[m] != e
assume a[m] < e
@post

@pre
assume l ≤ u
m := (l + u) div 2
assume a[m] != e
assume a[m] >= e
@post

Alternatively, most of you just used the slide from the lecture notes. This one also detailed the specifications themselves and dealt with the return variable.

b) Whenever some mathematical operation is not supported by a standard library, the solution is to implement it. The postcondition of this method for integer m(and the extra assume condition) is (l+u)/2 ≤ m < (l+u/2)+1