In section 3.10

REPLACE

```
      // Correct: IDLE=2’b00, XX=2’bx, S1=2’b01, S2=2’b10
      enum integer {IDLE, XX=’x, S1=2’b01, S2=2’b10} state, next;
```

WITH

```
      // Correct: IDLE=0, XX=’x, S1=1, S2=2
      enum integer {IDLE, XX=’x, S1=’b01, S2=’b10} state, next;
```

REPLACE

A sized constant can be used to set the size of the type. All sizes must be the same.

```
      // silver=4’h4, gold=4’h5 (all are 4 bits wide)
      enum {bronze=4’h3, silver, gold} medal4;
      // Syntax error: the width of the enum has been exceeded
      // in both of these examples
            enum {a=1'b0, b, c} alphabet;
            enum [0:0] {a,b,c} alphabet;
```

Any enumeration encoding value that is outside the representable range of the **enum** shall be an error.

Adding a constant range to the **enum** declaration can be used to set the size of the type. If any of the enum members are defined with a different sized constant, this shall be a syntax error.

```
      // Error in the bronze and gold member declarations
      enum bit [3:0] {bronze=5’h13, silver, gold=3’h5} medal4;
      // Correct declaration - bronze and gold sizes are redundant
      enum bit [3:0] {bronze=4’h13, silver, gold=4’h5} medal4;
```

WITH

Adding a constant range to the **enum** declaration can be used to set the size of the type. Any enumeration encoding value that is outside the representable range of the **enum** shall be an error. If any of the **enum** members are defined with a different sized constant, this shall be a syntax error.

```
      // Correct declaration - bronze and gold are unsized
      enum bit [3:0] {bronze=’h3, silver, gold=’h5} medal4;
      // Correct declaration - bronze and gold sizes are redundant
      enum bit [3:0] {bronze=4’h3, silver, gold=4’h5} medal4;
      // Error in the bronze and gold member declarations
      enum [3:0] {bronze=5’h13, silver, gold=3’h5} medal4;
      // Error in c declaration, requires at least 2 bits
      enum [0:0] {a,b,c} alphabet;
```