1.

```
module qOne
   input C1, C2, C3;
   wire [2:0]C;
   output reg out;
   always @(C,C1,C2,C3)
      C={C1,C2,C3)
      case (C)
         0: out=1;
         1: out=0;
               2: out=1;
               3: out=1;
               4: out=1;
               5: out=0;
               6: out=0;
               7: out=1;
      endcase
endmodule
```

2.

```
module qTwo
   input C1, C2;
   input [7:0] Y,X;
   wire [1:0] C;
   output reg [7:0] Z;
   always @(C,C1,C2)
      C={C1,C2}
      case (C)
      0: Z = X + Y;
         1: Z = X - Y;
         2: Z = X << Y;
         3: z =    X >> Y;
    endcase
endmodule
```

3.

```
module qThree
  input [7:0] X0, X3, X1, X2;
  input [2:0] C;
  output [7:0] out;
  wire[7:0] x, y;
  2to1Mux Stage0 (X0,X1,x,C[0]);
  2to1Mux Stage1 (X2,X3,y,C[1]);
  2to1Mux Stage2 (x,y,out,C[2]);
endmodule
```

4.

```
module FullAdder_4bit(Cin,X,Y,S,Cout)
  input Cin;
  input [3:0] X, Y,Z;
  output [3:0] S;
  output Cout;
  wire [3:1] c;
  FullAdder Stage0 (Cin,X[0],Y[0],C[1]);
  FullAdder Stage1 (C[1],X[1],Y[1],C[2]);
  FullAdder Stage2 (C[2],X[2],Y[2],C[3]);
  FullAdder Stage3 (C[3],X[3],Y[3],Cout);
```

```
endmodule



module qFour(Cin,X,Y,X,S,Cout)
  input Cin;
  input [3:0] X, Y,Z;
  output [3:0] S;
  output C;
  wire [3:0] carr;
  FullAdder_4bit Stage0 (Cin,X,Y,S,Carr);
  FullAdder_4bit Stage1 (Carr,Z,S,out,C);
endmodule
```