Q1) Write one function that copies a part of one int array a[ ] to another int array b[ ]. Assume the two arrays are of the same size. The call to the function would always contain array’s a[ ] and b[ ] as input. The other input(s) to the function could be:

- Only the begin position: for this case the function must copy values from the begin position of a[ ] to the last element of the array a[ ] to b[ ].
- Only the final position: for this case the function must copy from the first element of the array a[ ] to final position of a[ ] to b[ ].
- Both begin and final positions: copy elements between begin and final positions of a[ ] to b[ ].
- Neither: copy the entire array a[ ] onto b[ ].

Hint: Think default values.
Q2 a) What is the result of the following program?

```verilog
task Q2 (ref byte a[], reg[15:0] b, c, output u, v);
    reg [15:0] sum;
    b = c;
    sum = 0;
    foreach(a[i])begin
        sum = sum + a[i];
    end
    u = sum;
endtask

program test();

    initial begin
        reg[15:0] B = 10, C = 2, D = 4, E = 3;
        byte A[] = {2,3,4,5,6,0,1};
        Q2(A, B, C, D, E);
        foreach(A[i])
            $display(A[i]);
        $display(B, C, D, E);
    end
endprogram
```

b) Given the program:

```verilog
program test();

    initial begin
        a = 3; b = 4; c = 4; d = 6; e = 8;
        fork
            temp(a, b, c, d, e);
        begin
            wait(d); $display($time, "%d", d)
        end
    end
endprogram
```
What is the result of the above program for the following two definitions of the `temp()` task

i)
```plaintext
task temp (ref byte c, byte a, b, output e, d);
    c = a + b;
    #5;
    e = d + 2;
    #5;
    d = c + e;
    #5;
    d = c + d;
    #5;
endtask
```

ii)
```plaintext
task temp (ref byte c, byte a, b, d, output e);
    c = a + b;
    #5;
    e = d + 2;
    #5;
    d = c + e;
    #5;
    d = c + d;
    #5;
endtask
```
Q3) Consider the integration of modules A through D

Your task is to design an interface definition that would be used to connect these top level entities.

a) How many modport definitions would you require for this interface declaration?

b) Declare the interface for this type of system connectivity.

c) Code up the above top level integration of the above structure using the above interface declaration to stitch together the different modules.
Q4) Consider the following pieces of code:

```verilog
module accelerator(input clock, do, reg[3:0] addr, output reg done, reg [15:0] val );

...
reg [ 15:0] RAM [0:15];

always (@ posedge clock) begin
  if(do == 1)
    count = count + 1;
  else
    count = 0;
end

always @ (count or addr) begin
  if(count == 7) begin
    done = 1;
    val = RAM[addr];
  end
  else begin
    done = 0;
    val = 0;
  end
```

...
What potential downfall do you see in this code? Comment on the reason for this issue.