1. (15%)

(a) (5%) ROC $|z| > 2$: $h[n] = 2 \cdot (0.5)^n u[n] - 3 \cdot (-2)^n u[n]$.

(b) (5%) ROC $0.5 < |z| < 2$: $h[n] = 2 \cdot (0.5)^n u[n] + 3 \cdot (-2)^n u[-n-1]$.

(c) (5%) $\frac{11}{4} \cdot \frac{1-\frac{2}{11}z^{-1}}{(1-0.5z^{-1})(1+0.5z^{-1})}$.

2. (15%)

(a) (5%) $y_r(t) = \frac{3\sqrt{3}}{2\pi}\cos(2\pi \cdot 11 \cdot 10^3 \cdot t - \pi/3)$ since $\Omega T/2 = \pi/3$.

(b) (5%) $y_r(t) = \frac{3}{\pi}\cos(2\pi \cdot 10^3 \cdot t - \pi/6)$ since $\Omega T/2 = \pi/6$.

(c) (5%) $y_r(t) = \frac{2\sqrt{2}}{\pi}\cos(2\pi \cdot 44 \cdot t - 7\pi/1000)\cos(2\pi \cdot 11 \cdot 10^3 \cdot t - 3\pi/4)$ since $\Omega T/2 = \pi/4$. The digital carrier frequency $\omega_0 = \pi/2$. The phase delay is $4\omega_0^2/\pi^2 = 1$, and the group delay is $3 \cdot 4\omega_0^2/\pi^2 = 3$. So, the filtered signal $x_m[n] * h[n] = s[n-3]x[n-1]$. The reconstruction further causes another 0.5 samples of delay, so we have the terms of $\cos(2\pi \cdot 44 \cdot (t - 3.5/F_s))\cos(2\pi \cdot 11 \cdot 10^3 \cdot (t - 1.5/F_s))$.

3. (10%)

(a) (2%) $e^{-jk\pi/2}X[k] = \{1, 1-2j, -4+j, -1+8j, 16, -1-8j, -4+j, 1+2j\}$.

(b) (4%) $X[k]X^*[k] = \{1, 5, 17, 65, 256, 65, 17, 5\}$.

(c) (4%) $X_3[k] = X_R^{ce}[k] + jX_I^{co}[k] = \{1, 2+j, 4, 8+j, 16, 8-j, 4, 2-j\}$; $X_4[k] = X_I^{ce}[k] - jX_R^{co}[k] = \{0, 0, -1, 0, 0, 0, -1, 0\}$.

4. (15%)

(a) (4%) Skipped.

(b) (5%) Skipped.

(c) (6%) Skipped.

5. (15%)

(a) (3%) $X_c(j\Omega) = \frac{1}{\log(2/T) + j\Omega}$.

(b) (4%) $x[n] = 2^{-n}u[n]$ and $X(e^{j\omega}) = \frac{1}{1-0.5e^{-j\omega}}$. $X(e^{j\omega})$ is a frequency-aliased (and scaled) version of $X_c(j\Omega)$

(c) (4%) $x_1[n] = \frac{1}{1-2^{-4}} \cdot 2^{-n}$ where $n = 0, 1, 2, 3$.

(d) (4%) Perform 1024-point DFT on the zero-padded $x_1[n]$.

6. (10%) $A_s = 40$ dB and $\triangle\omega = 0.1\pi$.

(a) (3%) 3.3953.

(b) (3%) 0.55π.

(c) (4%) $L = 47$ (type-I); $M = 44.5$.

7. (20%)

(a) (10%)

```
x1_0 = [x1(1 : 256) zeros(1, 256)];
x1_1 = [x1(257 : 400) zeros(1, 112)];
x2_0 = [x2(1 : 256) zeros(1, 256)];
x2_1 = [x2(257 : 500) zeros(1, 12)];
X1_0 = myfft512(x1_0);
X1_1 = myfft512(x1_1);
X2_0 = myfft512(x2_0);
X2_1 = myfft512(x2_1);
X1_0_X2_0 = X1_0. * X2_0;
X1_1_X2_0 = X1_1. * X2_0;
X1_0_X2_1 = X1_0. * X2_1;
X1_1_X2_1 = X1_1. * X2_1;
x1_0_x2_0 = myfft512([X1_0_X2_0(1) X1_0_X2_0(512 : -1 : 2)])/512;
x1_1_x2_0 = myfft512([X1_1_X2_0(1) X1_1_X2_0(512 : -1 : 2)])/512;
x1_0_x2_1 = myfft512([X1_0_X2_1(1) X1_0_X2_1(512 : -1 : 2)])/512;
x1_1_x2_1 = myfft512([X1_1_X2_1(1) X1_1_X2_1(512 : -1 : 2)])/512;
x_m = x1_1_x2_0 + x1_0_x2_1;
x3 = [x1_0_x2_0(1 : 256) x1_0_x2_0(257 : 512) + x_m(1 : 256)
 x_m(257 : 512) + x1_1_x2_1(1 : 256) x1_1_x2_1(257 : 387)];
```

(1)

(b) (5%)

```
x4_upsample512 = [[x4 0]; zeros(7, 64)];
X4_upsample512 = myfft512(x4_upsample512(:)');
X4 = X4_upsample512(1 : 64);
X4_pad = [X4(1 : 32) X4(33)/2 zeros(1, 7 * 64 - 1) X4(33)/2 X4(34 : 64)];
X4_padflip = [X4_pad(1) X4_pad(512 : -1 : 2)];
x5 = myfft512(X4_padflip)/64;
x5 = x5(1 : 63 * 8);
```

(c) (5%)

```
x8 = x6 + j * x7;
X8 = myfft512(x8);
X8_real = real(X8);
X8_imag = imag(X8);
X8_real_ce = (X8_real + [X8_real(1)X8_real(512 : -1 : 2)])/2;
X8_real_co = (X8_real - [X8_real(1)X8_real(512 : -1 : 2)])/2;
X8_imag_ce = (X8_imag + [X8_imag(1)X8_imag(512 : -1 : 2)])/2;
X8_imag_co = (X8_imag - [X8_imag(1)X8_imag(512 : -1 : 2)])/2;
X6 = X8_real_ce + j * X8_imag_co;
X7 = X8_imag_ce - j * X8_real_co;
```