Final Results Table: (MATLAB code and resources in appendix)

	Q1	Q2	Q3
a)	0.9038	0.6402	0.4460
b)	0.9037	0.6405	N/A
c)	0.9037	0.6405	N/A

Appendix: Code used for all questions

Q1.a)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(1), f_x(1)];
p2 = [x(2), f_x(2)];
p3 = [x(3), f_x(3)];

%RHS, LHS and C
LHS = zeros(3);
LHS(1,1) = 1;
LHS(2,1) = p2(1) - p1(1);
LHS(2,2) = ((LHS(2,1))*2) + ((p3(1) -
p2(1))*2);
LHS(2,3) = p3(1) - p2(1);
LHS(3,3) = 1;

RHS = [0, 0, 0];
slope12 = (p2(2) - p1(2))/(p2(1) -
p1(1));
slope23 = (p3(2) - p2(2))/(p3(1) -
p2(1));
RHS(2) = (slope23 - slope12)*3;
RHS = RHS';

C = LHS\RHS;

%we will use s2 so solve for b2 & d2
d2 = (C(2+1) - C(2))/(3*(p3(1) -
p2(1)));
b2 = ((p3(2) - p2(2))/(p3(1) - p2(1)))
- ((p3(1) - p2(1))/3)*(2*C(2) + C(3));

%splines; we will use s2 for 0.23
s2 = p2(2) + b2*(0.23 - p2(1)) +
C(2)*(0.23 - p2(1))^2 + d2*(0.23 -
p2(1))^3;

display(s2);
```

Q1.b)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(1), f_x(1)];
p2 = [x(2), f_x(2)];
p3 = [x(3), f_x(3)];

%define f1 & f2 & f3
f_1_ = (p1(1) - p2(1))*(p1(1) - p3(1));
f_2_ = (p2(1) - p1(1))*(p2(1) - p3(1));
f_3_ = (p3(1) - p1(1))*(p3(1) - p2(1));
f_1 = (0.23 - p2(1))*(0.23 - p3(1));
f_2 = (0.23 - p1(1))*(0.23 - p3(1));
f_3 = (0.23 - p1(1))*(0.23 - p2(1));

%LaGrange
poly = p1(2)*(f_1/f_1_) +
p2(2)*(f_2/f_2_) + p3(2)*(f_3/f_3_);

display(poly);
```

Q1.c)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(1), f_x(1)];
p2 = [x(2), f_x(2)];
p3 = [x(3), f_x(3)];

%define coefficients a0, a1, a2
a_0 = p1(2);
a_1 = (p2(2) - p1(2))/(p2(1) - p1(1));
a_2 = ((p3(2) - p2(2))/(p3(1) - p2(1))
- a_1)/(p3(1) - p1(1));

%newton
poly = a_0 + a_1*(0.23 - p1(1)) +
a_2*(0.23 - p1(1))*(0.23 - p2(1));
display(poly);
```

Q2.a)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(4), f_x(4)];
p2 = [x(5), f_x(5)];
p3 = [x(6), f_x(6)];

%RHS, LHS and C
LHS = zeros(3);
LHS(1,1) = 1;
LHS(2,1) = p2(1) - p1(1);
LHS(2,2) = ((LHS(2,1))*2) + ((p3(1) -
p2(1))*2);
LHS(2,3) = p3(1) - p2(1);
LHS(3,3) = 1;

RHS = [0, 0, 0];
slope12 = (p2(2) - p1(2))/(p2(1) -
p1(1));
slope23 = (p3(2) - p2(2))/(p3(1) -
p2(1));
RHS(2) = (slope23 - slope12)*3;
RHS = RHS';

C = LHS\RHS;

%we will use s1 so solve for b1 & d1
d1 = (C(1+1) - C(1))/(3*(p2(1) -
p1(1)));
b1 = ((p2(2) - p1(2))/(p2(1) - p1(1)))
- ((p2(1) - p1(1))/3)*(2*C(1) + C(2));

%splines; we will use s1 for 0.78
s1 = p1(2) + b1*(0.78 - p1(1)) +
C(1)*(0.78 - p1(1))^2 + d1*(0.78 -
p1(1))^3;

display(s1);
```

Q2.b)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(1), f_x(1)];
p2 = [x(2), f_x(2)];
p3 = [x(3), f_x(3)];

%define f1 & f2 & f3
f_1_ = (p1(1) - p2(1))*(p1(1) - p3(1));
f_2_ = (p2(1) - p1(1))*(p2(1) - p3(1));
f_3_ = (p3(1) - p1(1))*(p3(1) - p2(1));
f_1 = (0.23 - p2(1))*(0.23 - p3(1));
f_2 = (0.23 - p1(1))*(0.23 - p3(1));
f_3 = (0.23 - p1(1))*(0.23 - p2(1));

%LaGrange
poly = p1(2)*(f_1/f_1_) +
p2(2)*(f_2/f_2_) + p3(2)*(f_3/f_3_);

display(poly);
```

Q2.c)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(1), f_x(1)];
p2 = [x(2), f_x(2)];
p3 = [x(3), f_x(3)];

%define coefficients a0, a1, a2
a_0 = p1(2);
a_1 = (p2(2) - p1(2))/(p2(1) - p1(1));
a_2 = ((p3(2) - p2(2))/(p3(1) - p2(1))
- a_1)/(p3(1) - p1(1));

%newton
poly = a_0 + a_1*(0.23 - p1(1)) +
a_2*(0.23 - p1(1))*(0.23 - p2(1));
display(poly);
```

Q3.a)

```
%arrays for x and f(x) values
x = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2,
1.4, 1.6, 1.8, 2, 2.2 ,2.4];

f_x = [1, 0.916, 0.836, 0.74, 0.624,
0.40, 0.224, 0.24, 0.265, 0.291, 0.316,
0.342, 0.368];

%get first three points
p1 = [x(11), f_x(11)];
p2 = [x(12), f_x(12)];
p3 = [x(13), f_x(13)];

%RHS, LHS and C
LHS = zeros(3);
LHS(1,1) = 1;
LHS(2,1) = p2(1) - p1(1);
LHS(2,2) = ((LHS(2,1))*2) + ((p3(1) -
p2(1))*2);
LHS(2,3) = p3(1) - p2(1);
LHS(3,3) = 1;

RHS = [0, 0, 0];
slope12 = (p2(2) - p1(2))/(p2(1) -
p1(1));
slope23 = (p3(2) - p2(2))/(p3(1) -
p2(1));
RHS(2) = (slope23 - slope12)*3;
RHS = RHS';

C = LHS\RHS;

%we will use s2 so solve for b2 & d2
d2 = (C(2+1) - C(2))/(3*(p3(1) -
p2(1)));
b2 = ((p3(2) - p2(2))/(p3(1) - p2(1)))
- ((p3(1) - p2(1))/3)*(2*C(2) + C(3));

%splines; we will use s2 for 3
s2 = p2(2) + b2*(3 - p2(1)) + C(2)*(3 -
p2(1))^2 + d2*(3 - p2(1))^3;

display(s2);
```

Helpful resources: (Popular results on Google & a Numerical Methods Book)

Book used for Cubic Splines:
http://mec.nit.ac.ir/file_part/master_doc/2015923203616561420321288.pdf

Tutorial used for Newton Polynomial:
https://nptel.ac.in/courses/122104019/numerical-analysis/Rathish-kumar/rathish-oct31/fratnode5.html

Post used for Lagrange Polynomial:
https://math.stackexchange.com/questions/523907/explanation-of-lagrange-interpolating-polynomial