1. Get copies of the sample program interp.c, and the makefile (Makefile) in one of your subdirectories. Compile and link the program by typing:

```
make interp < cr >
```

Run the program with the two datafiles interp_1.txt and interp_2.txt. Then pick a fourth degree polynomial of your choice. Pick five points off of its graph, put them in a data file and run the program on this data to see if it produces the original polynomial. Note that you will have to manually multiply the Newton form of the polynomial

\[
a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_k(x - x_0)(x - x_1)\ldots(x - x_{k-1})
\]

out to the usual form

\[
b_0 + b_1 x + b_2 x^2 + \ldots + b_k x^k
\]

to check equality.

2. Recursive functions are often useful in numerical work. The (not particularly efficient) divided_diff_1() in interp.c is one example. Get copies of the sample program interp_eff.c and make the executable and test on the same datafiles you used in part 1. Note that this program also produces the polynomial multiplied out in usual form:

\[
b_0 + b_1 x + b_2 x^2 + \ldots + b_k x^k
\]
as additional output. It does this with two functions, the first of which is recursive:

```
double kth_coef(int kth, int degree, double x[])
and
void newton2stand(int degree, double a[], double x[], double b[])
```

Check the source to set that you understand how recursion is being used in both these cases.