```
// Created by Kevin Shuman on 1/14/20.
// Phys_567_Homework_1.c
//
// The goal of this assignment is do a cubic interpolation of a given data set.
//
// The code below reproduces a txt file with just the data, no column names
// and generates a txt file with the interpolated data. It does this for each
// of the given set of data. One can then use these to make plots, which I use
 gnuplot.
//
//

// Compile with...
// gcc -w -o Phys_567_Homework_1 Phys_567_Homework_1.c -lgsl -lgslcblas -lm
//
// the "-w" flag gives all warning messages if there are any and "-o" allows me
// to define my file name, in this case "Phys_567_Homework_1". The others are
// just linkers to the librarys.
//
// Run with...
// ./Phys_567_Homework_1

// Below I'm including the typical C libraries and using GCC to compile my
 code,
// but I'm also including the GSL Spline library, which will allow me to do
 interpolation

// If you don't have the GSL libraries installed, you can install them by
 following the links
// MAC:
 http://connor-johnson
 .com/2014/10/12/using-the-gnu-scientific-library-on-a-mac/
// Linux: https://coral.ise.lehigh.edu/jild13/2016/07/11/hello/
// Windows: Good luck. I have done it, but I first found out how to install a
 linux termial on my computer and then somehow managed to get GSL. It was
 mostly luck, which is why I wish it.
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_spline.h>

// Here I'm defining my interpolating function, hence the name
// The input include a points which has the desired wavelengths you wish
// to find the reflectivity at, the file the data are in, and the name
// you want to give to the file with just the data in it. There is also a
// kill switch, with takes a 0 (off) or 1 (on), if you don't want to
// make a file of just the data.
double *Interpolation(double *input, FILE *in, int arraysize, char *name, int
 Kill);

void main()
{
```

```c
// I'm defining my pointers and allocating memory for them.
// The "size" is the length of the pointer, so changing that will affect
// the lengths of the pointers and for loops.
int size = 4000;
double *wave = (double *)malloc((size_t)sizeof(double)*(size));
double *ref = (double *)malloc((size_t)sizeof(double)*(size));

// Defining the file pointers here.
FILE *in, *out;
int i;

// It starts off with the below data set.
in = fopen("xray1727.dat.txt", "r");

// I'm creating the array of wavelengths I want to find the reflectivity of
// Notice I'm avoiding going outside the bounds of the data set.
// That's because interpolation outside those bounds is not interpolation.
for(i=0; i<size; i++) wave[i] = 0.01*(double)(i) + 1.0;

// Runs the function and generates a pointer of the reflectivity
ref = Interpolation(wave, in, size, "xray1727_data.txt", 1);

// Exports a data file containing the wavelength in the first column
// and the reflectivity in the second. Also, gives it the below name.
out = fopen("xray1727_interp.txt", "w");
for(i=0; i<size; i++) fprintf(out, "%.6e %.6e \n", wave[i], ref[i]);
fclose(out);

// Repeat this process for the other data sets.
//=====================================
in = fopen("xray8528.dat.txt", "r");

for(i=0; i<size; i++) wave[i] = 0.005*(double)(i) + 10.0;

ref = Interpolation(wave, in, size, "xray8528_data.txt", 1);

out = fopen("xray8528_interp.txt", "w");
for(i=0; i<size; i++) fprintf(out, "%.6e %.6e \n", wave[i], ref[i]);
fclose(out);

//=====================================
in = fopen("xray8529.dat.txt", "r");

for(i=0; i<size; i++) wave[i] = 0.005*(double)(i) + 10.0;

ref = Interpolation(wave, in, size, "xray8529_data.txt", 1);

out = fopen("xray8529_interp.txt", "w");
for(i=0; i<size; i++) fprintf(out, "%.6e %.6e \n", wave[i], ref[i]);
fclose(out);
```

```
    // I'm freeing the allocated memory here.
    free(wave); free(ref);

    // Lastly, I have the function just print to the terminal
    //==================================
    // I'm defining my pointers and allocating memory for them, again.
    size = 40;
    wave = (double *)malloc((size_t)sizeof(double)*(size));
    ref = (double *)malloc((size_t)sizeof(double)*(size));

    in = fopen("xray8529.dat.txt", "r");

    for(i=0; i<size; i++) wave[i] = 0.5*(double)(i) + 10.0;

    ref = Interpolation(wave, in, size, "xray8529_data.txt", 1);

    out = fopen("xray8529_interp.txt", "w");
    for(i=0; i<size; i++) printf("%.6e %.6e \n", wave[i], ref[i]);
    fclose(out);



    // I'm freeing the allocated memory here.
    free(wave); free(ref);
}


double *Interpolation(double *input, FILE *in, int arraysize, char *name, int
 Kill)
{
    // I first want to figure out how many lines there are in the file.
    // This will allow me to define pointers of the appropreate size.
    char ch;
    int numOfLines = 0;
    int i;
    FILE *out;

    // fgetc reads the first character and moves the position forward.
    // EOF is defined by the end of the file.
    // The while loop continues until the end of the file, but every time ch
     is '\n',
    // which is what defines the next line, we add one to our counter.
    while((ch=fgetc(in))!=EOF) if(ch=='\n') numOfLines += 1;


    // Just printing numOfLines to see what if it's correct.
    // printf("%i \n", numOfLines);

    // I want to define some pointers for the wavelength and reflectivity
    double *wave, *ref, *phase;
    wave = (double *)malloc((size_t)sizeof(double)*(numOfLines));
```

```
ref = (double *)malloc((size_t)sizeof(double)*(numOfLines));
phase = (double *)malloc((size_t)sizeof(double)*(numOfLines));

// I reset where my position is using rewind.
rewind(in);

// I then skip the first two lines, since they don't contain data.
fscanf(in, "%*[^\n]\n");
fscanf(in, "%*[^\n]\n");

// I then import the data into the arrays.
for(i=0; i<numOfLines-2; i++) fscanf(in, "%lf%lf%lf", &wave[i], &ref[i],
 &phase[i]);

// Then I close the file.
fclose(in);

if(Kill == 1)
{
    out = fopen(name, "w");
    for(i=0; i<numOfLines-2; i++) fprintf(out, "%.6e %.6e \n", wave[i],
     ref[i]);
    fclose(out);
}

// Printing the first ten lines of data to see if they are correct.
// for(i=0; i<numOfLines-2; i++) printf("%.6e %.6e \n", wave[i], ref[i]);

// Before I create my spline interpolation, I want to find the min and max
 of my wavelength
// so when I plug in values, the program warns me that I exceeded the
 bounds
double a, b, min, max;
min = INFINITY; // Just thow a huge initial value for the min.
max = 0.0; // We know the wavelength can't be zero.
for(i=0; i<numOfLines-3; i++)
{
    a = wave[i];
    b = wave[i+1];

    // Finding the maximum.
    if(a>b && a>=max) max = a;
    if(b>a && b>=max) max = b;

    // finding the minimum.
    if(a<b && a<=min) min = a;
    if(b<a && b<=min) min = b;
}

// Now for the interpolation.
// These variables and functions are defined in the GSL library here
```

```
    // https://www.gnu.org/software/gsl/doc/html/interp.html
    // In this, I'm doing a cubic spline interpolation, hence the spline
     everywhere.
    gsl_interp_accel *Waveacc = gsl_interp_accel_alloc();
    gsl_spline *Wavespline = gsl_spline_alloc (gsl_interp_cspline,
     numOfLines-2);
    gsl_spline_init(Wavespline, wave, ref, numOfLines-2);

    // Defining a pointer for the output, the relfectivity, and filling it.
    double *output;
    output = (double *)malloc((size_t)sizeof(double)*(arraysize));

    for(i=0; i<arraysize; i++) output[i] = gsl_spline_eval (Wavespline,
     input[i], Waveacc);

    // Freeing the allocated memory.
    free(wave); free(ref); free(phase);
    gsl_spline_free (Wavespline);
    gsl_interp_accel_free (Waveacc);

    return output;
}
```