This is the third and the final post on my series on writing a Powder XRD pattern (diffractogram) simulator from scratch using C.

Till now, we have seen how to calculate the atomic form factor and hence the structure factor.
This was needed for the calculation of the intensity of peaks in an X-ray diffraction pattern (diffractogram).

But that's not the only thing the intensity depends on. There is also something called the Lorentz-Polarization factor that depends on the value of theta.

Overall, intensity depends upon:
1. Multiplicity
2. Structure Factor
3. Lorentz-Polarization factor
4. Temperature
5. X-Ray Absorption
Out of these we can't really account for the last two factors in this simple program. This is standard practice in most of the softwares / tools.

So we'll use the following formula for calculating intensity for given atomic species, atomic positions, hkl values, and theta value.

$$I_{hkl} = M \times \frac{(1+\cos^2 2\theta)}{\sin^2\theta\cos\theta} \times |F_{hkl}|^2$$

where M is the multiplicity, $(1 + \cos^2 2\theta)$ is the Polarization factor and $\frac{1}{\sin^2\theta\cos\theta}$ is the Lorentz factor, and F_{hkl} is the structure factor.

The next thing we need to completely predict the XRD pattern is the peak positions, i.e. theta (2θ) values of peaks. This can be done by
1. running a loop on the possible hkl values.
2. Then using the lattice information to calculate the interplanar spacing, d(hkl).
3. Then use d_{hkl} to calculate theta using the Bragg's law.
4. Use theta and hkl values to calculate the real and imaginary part of structure factor as explained in the last post.
5. If structure factor is very small, i.e. around 0.01 then the intensity is going to be negligible, i.e. it won't show up as a peak. So you can reject such small structure factor values by rejecting the corresponding theta values. This would give you the peaks obtained in an XRD pattern.

Now due to a lot of equivalent reflections there would be several repetitions of theta (but not hkl). So you can use these repetitions to calculate the multiplicity as well as remove the repetitions for the output file. This can be done by finding the unique theta values and counting their occurrences.
Finally, to calculate the intensity, just square the structure factor magnitude and then multiply it by the Lorentz-Polarization factor and multiplicity, as already shown above.

Finally just store the output, i.e. the information of reflections, such as miller indices (hkl), theta/2theta, inter-planar spacing d, multiplicity, structure factor and intensity. Now although only information about unique reflections is needed to plot the pattern, but most softwares/tools such as VESTA, also provide information of the equivalent reflections. So the following program also generates two output files. One contains the information about all the reflections (including repetition). Then this data is processed, to find the information about repetitions and another file is generated that contains the data to be plotted.

Now, all the information above as well as in the last two posts should be sufficient to write your own code, but I am posting my own code too for reference and comparison.

One last thing you need to know about the following code is the input file structure.

The input file structure is really absurd and might annoy you. You're welcome to improve that part of the code. I am definitely gonna improve it soon.
But for now, let me explain the current expected structure.
The file should end in the .txt extension. That is absolutely necessary.
The first line should contain the number of atoms.
Then the second line contains the code corresponding to the lattice type. The codes are as follows:
1 for cubic, 2 for hexagonal, 3 for rhombohedral, 4 for tetragonal, 5 for orthorhombic, 6 for monoclinic and 7 for triclinic.

The lattice type would then decide what the next part of input looks like. If it's cubic (1), then the next line should have the one and only needed information, i.e. lattice parameter in Angstrom. If the lattice type is hexagonal (2) then we need lattice parameter a and c, so there will be two more lines giving this information. If lattice type is Monoclinic then there will be ____ lines giving the .

I know the input file structure is embarrassing to say the least, but I've included several examples in the end to make it easier to understand. Also pretty soon, I will be making a better input file structure as well as add support for CIF files.

So, the code is finally here:

```
/*XRD Pattern Simulator
By: Manas Sharma
mail: feedback@bragitoff.com
http://bragitoff.com
IG: @___physwhiz___
Forum: physwhiz.bragitoff.com
*/
#include<stdio.h>
#include<string.h>
#include<math.h>

/*
The following function takes the value of q(scattering vector) in the range 0 to 25
(Angstrom)^-1
and the name of the atomic specie using the atomic symbols
and returns the atomic form factor at that q value.
*/
double formFactorCalc(double q, char specie[]){
        //variable that will store the resulting form factor
        double result;
        int i, found=0,n;
        //Necessary tables needed for the calculations in array form
        char elements[211][10]={"H","H1-
","He","Li","Li1+","Be","Be2+","B","C","Cval","N","O","O1-","F","F1-
","Ne","Na","Na1+","Mg","Mg2+","Al","Al3+","Siv","Sival","Si","P","S","Cl","Cl1-
","Ar","K","K1+","Ca","Ca2+","Sc","Sc3+","Ti","Ti2+","Ti3+","Ti4+","V","V2+","V3+","V5+
","Cr","Cr2+","Cr3+","Mn","Mn2+","Mn3+","Mn4+","Fe","Fe2+","Fe3+","Co","Co2+","Co3+","N
i","Ni2+","Ni3+","Cu","Cu1+","Cu2+","Zn","Zn2+","Ga","Ga3+","Ge","Ge4+","As","Se","Br",
"Br1-
","Kr","Rb","Rb1+","Sr","Sr2+","Y","Y3+","Zr","Zr4+","Nb","Nb3+","Nb5+","Mo","Mo3+","Mo
5+","Mo6+","Tc","Ru","Ru3+","Ru4+","Rh","Rh3+","Rh4+","Pd","Pd2+","Pd4+","Ag","Ag1+","A
g2+","Cd","Cd2+","In","In3+","Sn","Sn2+","Sn4+","Sb","Sb3+","Sb5+","Te","I","I1-
","Xe","Cs","Cs1+","Ba","Ba2+","La","La3+","Ce","Ce3+","Ce4+","Pr","Pr3+","Pr4+","Nd","
Nd3+","Pm","Pm3+","Sm","Sm3+","Eu","Eu2+","Eu3+","Gd","Gd3+","Tb","Tb3+","Dy","Dy3+","H
```

```
o","Ho3+","Er","Er3+","Tm","Tm3+","Yb","Yb2+","Yb3+","Lu","Lu3+","Hf","Hf4+","Ta","Ta5+
","W","W6+","Re","Os","Os4+","Ir","Ir3+","Ir4+","Pt","Pt2+","Pt4+","Au","Au1+","Au3+","
Hg","Hg1+","Hg2+","Tl","Tl1+","Tl3+","Pb","Pb2+","Pb4+","Bi","Bi3+","Bi5+","Po","At","R
n","Fr","Ra","Ra2+","Ac","Ac3+","Th","Th4+","Pa","U","U3+","U4+","U6+","Np","Np3+","Np4
+","Np6+","Pu","Pu3+","Pu4+","Pu6+","Am","Cm","Bk","Cf"};
        double a1[]={0.489918 , 0.897661 , 0.8734 , 1.1282 , 0.6968 , 1.5919 , 6.2603 ,
2.0545 , 2.31 , 2.26069 , 12.2126 , 3.0485 , 4.1916 , 3.5392 , 3.6322 , 3.9553 , 4.7626
, 3.2565 , 5.4204 , 3.4988 , 6.4202 , 4.17448 , 6.2915 , 5.66269 , 4.43918 , 6.4345 ,
6.9053 , 11.4604 , 18.2915 , 7.4845 , 8.2186 , 7.9578 , 8.6266 , 15.6348 , 9.189 ,
13.4008 , 9.7595 , 9.11423 , 17.7344 , 19.5114 , 10.2971 , 10.106 , 9.43141 , 15.6887 ,
10.6406 , 9.54034 , 9.6809 , 11.2819 , 10.8061 , 9.84521 , 9.96253 , 11.7695 , 11.0424
, 11.1764 , 12.2841 , 11.2296 , 10.338 , 12.8376 , 11.4166 , 10.7806 , 13.338 , 11.9475
, 11.8168 , 14.0743 , 11.9719 , 15.2354 , 12.692 , 16.0816 , 12.9172 , 16.6723 ,
17.0006 , 17.1789 , 17.1718 , 17.3555 , 17.1784 , 17.5816 , 17.5663 , 18.0874 , 17.776
, 17.9268 , 17.8765 , 18.1668 , 17.6142 , 19.8812 , 17.9163 , 3.7025 , 21.1664 ,
21.0149 , 17.8871 , 19.1301 , 19.2674 , 18.5638 , 18.5003 , 19.2957 , 18.8785 , 18.8545
, 19.3319 , 19.1701 , 19.2493 , 19.2808 , 19.1812 , 19.1643 , 19.2214 , 19.1514 ,
19.1624 , 19.1045 , 19.1889 , 19.1094 , 18.9333 , 19.6418 , 18.9755 , 19.8685 , 19.9644
, 20.1472 , 20.2332 , 20.2933 , 20.3892 , 20.3524 , 20.3361 , 20.1807 , 20.578 ,
20.2489 , 21.1671 , 20.8036 , 20.3235 , 22.044 , 21.3727 , 20.9413 , 22.6845 , 21.961 ,
23.3405 , 22.5527 , 24.0042 , 23.1504 , 24.6274 , 24.0063 , 23.7497 , 25.0709 , 24.3466
, 25.8976 , 24.9559 , 26.507 , 25.5395 , 26.9049 , 26.1296 , 27.6563 , 26.722 , 28.1819
, 27.3083 , 28.6641 , 28.1209 , 27.8917 , 28.9476 , 28.4628 , 29.144 , 28.8131 ,
29.2024 , 29.1587 , 29.0818 , 29.4936 , 28.7621 , 28.1894 , 30.419 , 27.3049 , 30.4156
, 30.7058 , 27.0059 , 29.8429 , 30.9612 , 16.8819 , 28.0109 , 30.6886 , 20.6809 ,
25.0853 , 29.5641 , 27.5446 , 21.3985 , 30.8695 , 31.0617 , 21.7886 , 32.1244 , 33.3689
, 21.8053 , 33.5364 , 34.6726 , 35.3163 , 35.5631 , 35.9299 , 35.763 , 35.215 , 35.6597
, 35.1736 , 35.5645 , 35.1007 , 35.8847 , 36.0228 , 35.5747 , 35.3715 , 34.8509 ,
36.1874 , 35.7074 , 35.5103 , 35.0136 , 36.5254 , 35.84 , 35.6493 , 35.1736 , 36.6706 ,
36.6488 , 36.7881 , 36.9185};
        double a2[]={0.262003 , 0.565616 , 0.6309 , 0.7508 , 0.7888 , 1.1278 , 0.8849 ,
1.3326 , 1.02 , 1.56165 , 3.1322 , 2.2868 , 1.63969 , 2.6412 , 3.51057 , 3.1125 ,
3.1736 , 3.9362 , 2.1735 , 3.8378 , 1.9002 , 3.3876 , 3.0353 , 3.07164 , 3.20345 ,
4.1791 , 5.2034 , 7.1964 , 7.2084 , 6.7723 , 7.4398 , 7.4917 , 7.3873 , 7.9518 , 7.3679
, 8.0273 , 7.3558 , 7.62174 , 8.73816 , 8.23473 , 7.3511 , 7.3541 , 7.7419 , 8.14208 ,
7.3537 , 7.7509 , 7.81136 , 7.3573 , 7.362 , 7.87194 , 7.97057 , 7.3573 , 7.374 ,
7.3863 , 7.3409 , 7.3883 , 7.88173 , 7.292 , 7.4005 , 7.75868 , 7.1676 , 7.3573 ,
7.11181 , 7.0318 , 7.3862 , 6.7006 , 6.69883 , 6.3747 , 6.70003 , 6.0701 , 5.8196 ,
5.2358 , 6.3338 , 6.7286 , 9.6435 , 7.6598 , 9.8184 , 8.1373 , 10.2946 , 9.1531 ,
10.948 , 10.0562 , 12.0144 , 18.0653 , 13.3417 , 17.2356 , 18.2017 , 18.0992 , 11.175 ,
11.0948 , 12.9182 , 13.2885 , 13.1787 , 14.3501 , 14.1259 , 13.9806 , 15.5017 , 15.2096
, 14.79 , 16.6885 , 15.9719 , 16.2456 , 17.6444 , 17.2535 , 18.5596 , 18.1108 , 19.1005
, 19.0548 , 19.7131 , 19.0455 , 18.933 , 19.0302 , 19.0138 , 18.9949 , 18.997 , 19.0298
, 19.1062 , 19.1278 , 19.297 , 19.1136 , 19.599 , 19.3763 , 19.7695 , 19.559 , 19.8186
, 19.6697 , 19.7491 , 20.0539 , 19.6847 , 19.9339 , 19.6095 , 20.1108 , 19.4258 ,
20.2599 , 19.0886 , 19.9504 , 20.3745 , 19.0798 , 20.4208 , 18.2185 , 20.3271 , 17.6383
, 20.2861 , 17.294 , 20.0994 , 16.4285 , 19.7748 , 15.8851 , 19.332 , 15.4345 , 17.6817
, 18.7614 , 15.2208 , 18.121 , 15.1726 , 18.4601 , 15.2293 , 18.8407 , 15.43 , 19.3763
, 15.7189 , 16.155 , 15.2637 , 16.7296 , 15.862 , 15.5512 , 17.7639 , 16.7224 , 15.9829
, 18.5913 , 17.8204 , 16.9029 , 19.0417 , 18.4973 , 18.06 , 19.1584 , 20.4723 , 18.3481
, 13.0637 , 19.5682 , 18.8003 , 12.951 , 19.5026 , 25.0946 , 15.4733 , 19.0211 ,
21.2816 , 23.0547 , 22.9064 , 21.67 , 23.1032 , 22.1112 , 23.4219 , 22.4418 , 23.2948 ,
23.4128 , 22.5259 , 22.5326 , 22.7584 , 23.5964 , 22.613 , 22.5787 , 22.7286 , 23.8083
, 22.7169 , 22.646 , 22.7181 , 24.0992 , 24.4096 , 24.7736 , 25.1995};
```

```
        double a3[]={0.196767 , 0.415815 , 0.3112 , 0.6175 , 0.3414 , 0.5391 , 0.7993 ,
1.0979 , 1.5886 , 1.05075 , 2.0125 , 1.5463 , 1.52673 , 1.517 , 1.26064 , 1.4546 ,
1.2674 , 1.3998 , 1.2269 , 1.3284 , 1.5936 , 1.20296 , 1.9891 , 2.62446 , 1.19453 ,
1.78 , 1.4379 , 6.2556 , 6.5337 , 0.6539 , 1.0519 , 6.359 , 1.5899 , 8.4372 , 1.6409 ,
1.65943 , 1.6991 , 2.2793 , 5.25691 , 2.01341 , 2.0703 , 2.2884 , 2.15343 , 2.03081 ,
3.324 , 3.58274 , 2.87603 , 3.0193 , 3.5268 , 3.56531 , 2.76067 , 3.5222 , 4.1346 ,
3.3948 , 4.0034 , 4.7393 , 4.76795 , 4.4438 , 5.3442 , 5.22746 , 5.6158 , 6.2455 ,
5.78135 , 5.1652 , 6.4668 , 4.3591 , 6.06692 , 3.7068 , 6.06791 , 3.4313 , 3.9731 ,
5.6377 , 5.5754 , 5.5493 , 5.1399 , 5.8981 , 5.422 , 2.5654 , 5.72629 , 1.76795 ,
5.41732 , 1.01118 , 4.04183 , 11.0177 , 10.799 , 12.8876 , 11.7423 , 11.4632 , 6.57891
, 4.64901 , 4.86337 , 9.32602 , 4.71304 , 4.73425 , 3.32515 , 2.53464 , 5.29537 ,
4.32234 , 2.89289 , 4.8045 , 5.27475 , 4.3709 , 4.461 , 4.47128 , 4.2948 , 3.78897 ,
4.4585 , 4.5648 , 3.4182 , 5.0371 , 5.10789 , 2.41253 , 6.14487 , 7.5138 , 7.8069 ,
8.9767 , 10.662 , 10.2821 , 10.888 , 10.9054 , 11.3727 , 11.6323 , 11.8513 , 11.9369 ,
12.1233 , 12.3856 , 12.1329 , 12.4668 , 12.774 , 12.12 , 13.1235 , 12.0671 , 13.4396 ,
11.9202 , 13.7603 , 11.8034 , 11.8509 , 13.8518 , 11.8708 , 14.3167 , 12.2471 , 14.5596
, 11.9812 , 14.5583 , 11.9788 , 14.9779 , 12.1506 , 15.1542 , 12.3339 , 15.3087 ,
13.3335 , 12.6072 , 15.1 , 12.8429 , 14.7586 , 12.7285 , 14.5135 , 12.8268 , 14.4327 ,
13.0544 , 14.5564 , 14.9305 , 14.7458 , 15.6115 , 13.6145 , 14.2326 , 15.7131 , 13.2153
, 13.7348 , 25.5582 , 14.3359 , 12.7801 , 21.6575 , 16.8883 , 12.8374 , 15.538 ,
18.7478 , 11.9328 , 18.442 , 19.1406 , 12.0175 , 16.5877 , 19.1053 , 19.2497 , 13.1138
, 9.49887 , 8.0037 , 12.1439 , 12.4739 , 7.91342 , 12.5977 , 8.19216 , 12.7473 ,
9.78554 , 14.1891 , 14.9491 , 12.2165 , 12.0291 , 14.0099 , 15.6402 , 12.9898 , 12.7766
, 14.3884 , 16.7707 , 13.5807 , 13.3595 , 14.7635 , 17.3415 , 17.399 , 17.8919 ,
18.3317};
        double a4[]={0.049879 , 0.116973 , 0.178 , 0.4653 , 0.1563 , 0.7029 , 0.1647 ,
0.7068 , 0.865 , 0.839259 , 1.1663 , 0.867 , -20.307 , 1.0243 , 0.940706 , 1.1251 ,
1.1128 , 1.0032 , 2.3073 , 0.8497 , 1.9646 , 0.528137 , 1.541 , 1.3932 , 0.41653 ,
1.4908 , 1.5863 , 1.6455 , 2.3386 , 1.6442 , 0.8659 , 1.1915 , 1.0211 , 0.8537 , 1.468
, 1.57936 , 1.9021 , 0.087899 , 1.92134 , 1.5208 , 2.0571 , 0.0223 , 0.016865 , -9.576
, 1.4922 , 0.509107 , 0.113575 , 2.2441 , 0.2184 , 0.323613 , 0.054447 , 2.3045 ,
0.4399 , 0.0724 , 2.3488 , 0.7108 , 0.725591 , 2.38 , 0.9773 , 0.847114 , 1.6735 ,
1.5578 , 1.14523 , 2.41 , 1.394 , 2.9623 , 1.0066 , 3.683 , 0.859041 , 4.2779 , 4.3543
, 3.9851 , 3.7272 , 3.5375 , 1.5292 , 2.7817 , 2.6694 , -34.193 , 3.26588 , -33.108 ,
3.65721 , -2.6479 , 3.53346 , 1.94715 , 0.337905 , 3.7429 , 2.30951 , 0.740625 , 0 ,
2.71263 , 1.56756 , 3.00964 , 2.18535 , 1.28918 , -6.1989 , -5.6526 , 0.605844 , 0 ,
-7.9492 , 1.0463 , 0.357534 , 0 , 1.6029 , 0 , 2.0396 , 0 , 2.4663 , 0.487 , 0.0193 ,
2.6827 , 0.288753 , 0 , 2.5239 , 2.2735 , 2.8868 , 1.99 , 1.4953 , 0.9615 , 2.6959 ,
0.77634 , 3.28719 , 0.336048 , 3.33049 , 0.612376 , 0.144583 , 2.82428 , 0.97518 ,
0.296689 , 2.85137 , 1.51031 , 2.87516 , 2.07492 , 2.89604 , 2.71488 , 2.9227 , 3.87243
, 3.26503 , 3.54545 , 3.7149 , 2.95354 , 3.773 , 2.96577 , 4.50073 , 3.63837 , 4.93676
, 2.98233 , 5.17379 , 2.98706 , 5.38348 , 2.98963 , 5.14657 , 5.47647 , 3.71601 ,
5.59415 , 4.30013 , 5.59927 , 4.76492 , 5.38695 , 5.11982 , 5.06412 , 5.44174 , 5.67589
, 5.06795 , 5.83377 , 5.82008 , 5.53672 , 5.7837 , 6.35234 , 5.92034 , 5.86 , 6.58077 ,
6.52354 , 5.9676 , 6.48216 , 6.89912 , 5.52593 , 6.82847 , 7.00574 , 5.9696 , 7.01107 ,
6.96886 , 6.4692 , 7.10295 , 6.91555 , 7.02588 , 7.42518 , 7.4433 , 2.11253 , 3.21097 ,
7.65078 , 4.08655 , 7.05545 , 4.80703 , 5.29444 , 4.17287 , 4.188 , 5.37073 , 4.7984 ,
1.21457 , 4.1855 , 5.43227 , 4.92159 , 1.75669 , 3.47947 , 5.66016 , 5.18831 , 2.28678
, 3.49331 , 4.21665 , 4.23284 , 4.24391};
        double b1[]={20.6593 , 53.1368 , 9.1037 , 3.9546 , 4.6237 , 43.6427 , 0.0027 ,
23.2185 , 20.8439 , 22.6907 , 0.0057 , 13.2771 , 12.8573 , 10.2825 , 5.27756 , 8.4042 ,
3.285 , 2.6671 , 2.8275 , 2.1676 , 3.0387 , 1.93816 , 2.4386 , 2.6652 , 1.64167 ,
1.9067 , 1.4679 , 0.0104 , 0.0066 , 0.9072 , 12.7949 , 12.6331 , 10.4421 , -0.0074 ,
9.0213 , 0.29854 , 7.8508 , 7.5243 , 0.22061 , 0.178847 , 6.8657 , 6.8818 , 6.39535 ,
```

```
0.679003 , 6.1038 , 5.66078 , 5.59463 , 5.3409 , 5.2796 , 4.91797 , 4.8485 , 4.7611 ,
4.6538 , 4.6147 , 4.2791 , 4.1231 , 3.90969 , 3.8785 , 3.6766 , 3.5477 , 3.5828 ,
3.3669 , 3.37484 , 3.2655 , 2.9946 , 3.0669 , 2.81262 , 2.8509 , 2.53718 , 2.6345 ,
2.4098 , 2.1723 , 2.2059 , 1.9384 , 1.7888 , 1.7139 , 1.5564 , 1.4907 , 1.4029 ,
1.35417 , 1.27618 , 1.2148 , 1.18865 , 0.019175 , 1.12446 , 0.2772 , 0.014734 ,
0.014345 , 1.03649 , 0.864132 , 0.80852 , 0.847329 , 0.844582 , 0.751536 , 0.764252 ,
0.760825 , 0.698655 , 0.696219 , 0.683839 , 0.6446 , 0.646179 , 0.645643 , 0.5946 ,
0.597922 , 0.5476 , 0.551522 , 5.8303 , 0.5036 , 5.764 , 5.3034 , 0.467196 , 5.44853 ,
4.81742 , 4.347 , 4.3579 , 3.9282 , 3.569 , 3.552 , 3.216 , 3.21367 , 2.94817 , 2.9207
, 2.81219 , 2.77691 , 2.65941 , 2.77393 , 2.6452 , 2.54467 , 2.66248 , 2.52722 , 2.5627
, 2.4174 , 2.47274 , 2.31641 , 2.3879 , 2.27783 , 2.22258 , 2.25341 , 2.13553 , 2.24256
, 2.05601 , 2.1802 , 1.9804 , 2.07051 , 1.91072 , 2.07356 , 1.84659 , 2.02859 , 1.78711
, 1.9889 , 1.78503 , 1.73272 , 1.90182 , 1.68216 , 1.83262 , 1.59136 , 1.77333 ,
1.50711 , 1.72029 , 1.42755 , 1.67191 , 1.62903 , 1.37113 , 1.59279 , 1.34323 , 1.30923
, 1.51293 , 1.32927 , 1.24813 , 0.4611 , 1.35321 , 1.2199 , 0.545 , 1.39507 , 1.21152 ,
0.65515 , 1.4711 , 1.1008 , 0.6902 , 1.3366 , 1.00566 , 0.704 , 1.2356 , 0.91654 ,
0.700999 , 0.68587 , 0.6631 , 0.646453 , 0.616341 , 0.604909 , 0.589092 , 0.579689 ,
0.563359 , 0.555054 , 0.547751 , 0.5293 , 0.52048 , 0.516598 , 0.507079 , 0.511929 ,
0.502322 , 0.498626 , 0.48981 , 0.499384 , 0.484938 , 0.481422 , 0.473204 , 0.483629 ,
0.465154 , 0.451018 , 0.437533};
        double b2[]={7.74039 , 15.187 , 3.3568 , 1.0524 , 1.9557 , 1.8623 , 0.8313 ,
1.021 , 10.2075 , 0.656665 , 9.8933 , 5.7011 , 4.17236 , 4.2944 , 14.7353 , 3.4262 ,
8.8422 , 6.1153 , 79.2611 , 4.7542 , 0.7426 , 4.14553 , 32.3337 , 38.6634 , 3.43757 ,
27.157 , 22.2151 , 1.1662 , 1.1717 , 14.8407 , 0.7748 , 0.7674 , 0.6599 , 0.6089 ,
0.5729 , 7.9629 , 0.5 , 0.457585 , 7.04716 , 6.67018 , 0.4385 , 0.4409 , 0.383349 ,
5.40135 , 0.392 , 0.344261 , 0.334393 , 0.3432 , 0.3435 , 0.294393 , 0.283303 , 0.3072
, 0.3053 , 0.3005 , 0.2784 , 0.2726 , 0.238668 , 0.2565 , 0.2449 , 0.22314 , 0.247 ,
0.2274 , 0.244078 , 0.2333 , 0.2031 , 0.2412 , 0.22789 , 0.2516 , 0.205855 , 0.2647 ,
0.2726 , 16.5796 , 19.3345 , 16.5623 , 17.3151 , 14.7957 , 14.0988 , 12.6963 , 12.8006
, 11.2145 , 11.916 , 10.1483 , 11.766 , 1.13305 , 0.028781 , 1.0958 , 1.03031 , 1.02238
, 8.48061 , 8.14487 , 8.43467 , 8.37164 , 8.12534 , 8.21758 , 7.84438 , 7.62436 ,
7.98929 , 7.55573 , 7.14833 , 7.4726 , 7.19123 , 7.18544 , 6.9089 , 6.80639 , 6.3776 ,
6.3247 , 0.5031 , 5.8378 , 0.4655 , 0.4607 , 5.22126 , 0.467973 , 0.420885 , 0.3814 ,
0.3815 , 0.344 , 0.3107 , 0.3086 , 0.2756 , 0.28331 , 0.244475 , 0.250698 , 0.226836 ,
0.23154 , 0.21885 , 0.222087 , 0.214299 , 0.202481 , 0.210628 , 0.199237 , 0.202088 ,
0.185769 , 0.196451 , 0.174081 , 0.1942 , 0.17353 , 0.16394 , 0.181951 , 0.155525 ,
0.196143 , 0.149525 , 0.202172 , 0.143384 , 0.19794 , 0.139358 , 0.223545 , 0.13729 ,
0.238849 , 0.136974 , 0.257119 , 0.15997 , 0.13879 , 9.98519 , 0.142292 , 9.5999 ,
0.128903 , 9.37046 , 0.116741 , 9.2259 , 0.104621 , 9.09227 , 8.97948 , 6.84706 ,
8.86553 , 7.10909 , 6.71983 , 8.81174 , 7.38979 , 6.60834 , 8.6216 , 7.7395 , 6.82872 ,
8.4484 , 7.65105 , 7.05639 , 8.70751 , 0.517394 , 6.53852 , 2.3576 , 0.488383 , 6.10926
, 2.9238 , 6.24149 , 0.39042 , 3.55078 , 3.97458 , 4.0691 , 4.17619 , 3.87135 , 3.5767
, 3.65155 , 3.41437 , 3.46204 , 3.24498 , 3.41519 , 3.3253 , 3.12293 , 3.05053 , 2.8903
, 3.25396 , 3.03807 , 2.96627 , 2.81099 , 3.26371 , 2.96118 , 2.8902 , 2.73848 ,
3.20647 , 3.08997 , 3.04619 , 3.00775};
        double b3[]={49.5519 , 186.576 , 22.9276 , 85.3905 , 0.6316 , 103.483 , 2.2758
, 60.3498 , 0.5687 , 9.75618 , 28.9975 , 0.3239 , 47.0179 , 0.2615 , 0.442258 , 0.2306
, 0.3136 , 0.2001 , 0.3808 , 0.185 , 31.5472 , 0.228753 , 0.6785 , 0.916946 , 0.2149 ,
0.526 , 0.2536 , 18.5194 , 19.5424 , 43.8983 , 213.187 , -0.002 , 85.7484 , 10.3116 ,
136.108 , -0.28604 , 35.6338 , 19.5361 , -0.15762 , -0.29263 , 26.8938 , 20.3004 ,
15.1908 , 9.97278 , 20.2626 , 13.3075 , 12.8288 , 17.8674 , 14.343 , 10.8171 , 10.4852
, 15.3535 , 12.0546 , 11.6729 , 13.5359 , 10.2443 , 8.35583 , 12.1763 , 8.873 , 7.64468
, 11.3966 , 8.6625 , 7.9876 , 10.3163 , 7.0826 , 10.7805 , 6.36441 , 11.4468 , 5.47913
, 12.9479 , 15.2372 , 0.2609 , 0.2871 , 0.2261 , 0.2748 , 0.1603 , 0.1664 , 24.5651 ,
```

```
0.125599 , 22.6599 , 0.117622 , 21.6054 , 0.204785 , 10.1621 , 9.28206 , 11.004 ,
9.53659 , 8.78809 , 0.058881 , 21.5707 , 24.7997 , 0.017662 , 0.36495 , 25.8749 ,
21.2487 , 19.3317 , 25.2052 , 22.5057 , 17.9144 , 24.6605 , 21.7326 , 21.4072 , 24.7008
, 20.2521 , 25.8499 , 17.3595 , 26.8909 , 23.3752 , 14.0049 , 27.9074 , 19.5902 ,
14.1259 , 28.5284 , 27.766 , 29.5259 , 26.4659 , 24.3879 , 23.7128 , 20.2073 , 20.0558
, 18.7726 , 17.8211 , 17.6083 , 16.5408 , 15.7992 , 16.7669 , 15.323 , 14.8137 , 15.885
, 14.1783 , 15.1009 , 13.1275 , 14.3996 , 12.1571 , 13.7546 , 11.6096 , 11.311 ,
12.9331 , 10.5782 , 12.6648 , 10.0499 , 12.1899 , 9.34972 , 11.4407 , 8.80018 , 11.3604
, 8.36225 , 10.9975 , 7.96778 , 10.6647 , 8.18304 , 7.64412 , 0.261033 , 7.33727 ,
0.275116 , 6.76232 , 0.295977 , 6.31524 , 0.321703 , 5.93667 , 0.3505 , 0.382661 ,
0.165191 , 0.417916 , 0.204633 , 0.167252 , 0.424593 , 0.263297 , 0.16864 , 1.4826 ,
0.356752 , 0.212867 , 1.5729 , 0.443378 , 0.284738 , 1.96347 , 7.43463 , 0.219074 ,
8.618 , 6.7727 , 0.147041 , 8.7937 , 0.469999 , 5.71414 , 9.55642 , 11.3824 , 14.0422 ,
23.1052 , 19.9887 , 12.601 , 18.599 , 12.9187 , 17.8309 , 13.4661 , 16.9235 , 16.0927 ,
12.7148 , 12.5723 , 13.1767 , 15.3622 , 12.1449 , 11.9484 , 12.33 , 14.9455 , 11.5331 ,
11.316 , 11.553 , 14.3136 , 13.4346 , 12.8946 , 12.4044};
        double b4[]={2.20159 , 3.56709 , 0.9821 , 168.261 , 10.0953 , 0.542 , 5.1146 ,
0.1403 , 51.6512 , 55.5949 , 0.5826 , 32.9089 , -0.01404 , 26.1476 , 47.3437 , 21.7184
, 129.424 , 14.039 , 7.1937 , 10.1411 , 85.0886 , 8.28524 , 81.6937 , 93.5458 , 6.65365
, 68.1645 , 56.172 , 47.7784 , 60.4486 , 33.3929 , 41.6841 , 31.9128 , 178.437 ,
25.9905 , 51.3531 , 16.0662 , 116.105 , 61.6558 , 15.9768 , 12.9464 , 102.478 , 115.122
, 63.969 , 0.940464 , 98.7399 , 32.4224 , 32.8761 , 83.7543 , 41.3235 , 24.1281 ,
27.573 , 76.8805 , 31.2809 , 38.5566 , 71.1692 , 25.6466 , 18.3491 , 66.3421 , 22.1626
, 16.9673 , 64.8126 , 25.8487 , 19.897 , 58.7097 , 18.0995 , 61.4135 , 14.4122 ,
54.7625 , 11.603 , 47.7972 , 43.8163 , 41.4328 , 58.1535 , 39.3972 , 164.934 , 31.2087
, 132.376 , -0.0138 , 104.354 , -0.01319 , 87.6627 , -0.10276 , 69.7957 , 28.3389 ,
25.7228 , 61.6584 , 26.6307 , 23.3452 , 0 , 86.8472 , 94.2928 , 22.887 , 20.8504 ,
98.6062 , -0.01036 , -0.0102 , 76.8986 , 0 , 0.005127 , 99.8156 , 66.1147 , 0 , 87.4825
, 0 , 92.8029 , 0 , 83.9571 , 62.2061 , -0.7583 , 75.2825 , 55.5113 , 0 , 70.8403 ,
66.8776 , 84.9304 , 64.2658 , 213.904 , 59.4565 , 167.202 , 51.746 , 133.124 , 54.9453
, 127.113 , 43.1692 , 62.2355 , 143.644 , 36.4065 , 45.4643 , 137.903 , 30.8717 ,
132.721 , 27.4491 , 128.007 , 24.8242 , 123.174 , 26.5156 , 22.9966 , 101.398 , 21.7029
, 115.362 , 21.2773 , 111.874 , 19.581 , 92.6566 , 18.5908 , 105.703 , 17.8974 ,
102.961 , 17.2922 , 100.417 , 20.39 , 16.8153 , 84.3298 , 16.3535 , 72.029 , 14.0366 ,
63.3644 , 12.4244 , 57.056 , 11.1972 , 52.0861 , 48.1647 , 18.003 , 45.0011 , 20.3254 ,
17.4911 , 38.6103 , 22.9426 , 16.9392 , 36.3956 , 26.4043 , 18.659 , 38.3246 , 28.2262
, 20.7482 , 45.8149 , 28.8482 , 17.2114 , 47.2579 , 23.8132 , 14.714 , 48.0093 ,
20.3185 , 12.8285 , 47.0045 , 45.4715 , 44.2473 , 150.645 , 142.325 , 29.8436 , 117.02
, 25.9443 , 99.1722 , 23.9533 , 105.251 , 100.613 , 26.3394 , 23.4582 , 25.2017 ,
97.4908 , 25.4928 , 22.7502 , 22.6581 , 105.98 , 24.3992 , 21.8301 , 20.9303 , 102.273
, 88.4834 , 86.003 , 83.7881};
        double c[]={0.001305 , 0.002389 , 0.0064 , 0.0377 , 0.0167 , 0.0385 , -6.1092 ,
-0.1932 , 0.2156 , 0.286977 , -11.529 , 0.2508 , 21.9412 , 0.2776 , 0.653396 , 0.3515 ,
0.676 , 0.404 , 0.8584 , 0.4853 , 1.1151 , 0.706786 , 1.1407 , 1.24707 , 0.746297 ,
1.1149 , 0.8669 , -9.5574 , -16.378 , 1.4445 , 1.4228 , -4.9978 , 1.3751 , -14.875 ,
1.3329 , -6.6667 , 1.2807 , 0.897155 , -14.652 , -13.28 , 1.2199 , 1.2298 , 0.656565 ,
1.7143 , 1.1832 , 0.616898 , 0.518275 , 1.0896 , 1.0874 , 0.393974 , 0.251877 , 1.0369
, 1.0097 , 0.9707 , 1.0118 , 0.9324 , 0.286667 , 1.0341 , 0.8614 , 0.386044 , 1.191 ,
0.89 , 1.14431 , 1.3041 , 0.7807 , 1.7189 , 1.53545 , 2.1313 , 1.45572 , 2.531 , 2.8409
, 2.9557 , 3.1776 , 2.825 , 3.4873 , 2.0782 , 2.5064 , 41.4025 , 1.91213 , 40.2602 ,
2.06929 , 9.41454 , 3.75591 , -12.912 , -6.3934 , 4.3875 , -14.421 , -14.316 , 0.344941
, 5.40428 , 5.37874 , -3.1892 , 1.42357 , 5.328 , 11.8678 , 11.2835 , 5.26593 , 5.2916
, 13.0174 , 5.179 , 5.21572 , 5.21404 , 5.0694 , 5.11937 , 4.9391 , 4.99635 , 4.7821 ,
4.7861 , 3.9182 , 4.5909 , 4.69626 , 4.69263 , 4.352 , 4.0712 , 4.0714 , 3.7118 ,
```

```
3.3352 , 3.2791 , 2.7731 , 3.02902 , 2.14678 , 2.4086 , 1.86264 , 2.09013 , 1.5918 ,
2.0583 , 1.77132 , 1.24285 , 1.98486 , 1.47588 , 2.02876 , 1.19499 , 2.20963 , 0.954586
, 2.5745 , 1.36389 , 0.759344 , 2.4196 , 0.645089 , 3.58324 , 0.691967 , 4.29728 ,
0.68969 , 4.56796 , 0.852795 , 5.92046 , 1.17613 , 6.75621 , 1.63929 , 7.56672 ,
3.70983 , 2.26001 , 7.97628 , 2.97573 , 8.58154 , 2.39699 , 9.24354 , 1.78555 , 9.8875
, 1.01074 , 10.472 , 11.0005 , 6.49804 , 11.4722 , 8.27903 , 6.96824 , 11.6883 ,
9.85329 , 7.39534 , 12.0658 , 11.2299 , 9.0968 , 12.6089 , 12.0205 , 10.6268 , 13.1746
, 12.5258 , 9.8027 , 13.4118 , 12.4734 , 8.08428 , 13.5782 , 12.4711 , -6.7994 , 13.677
, 13.7108 , 13.6905 , 13.7247 , 13.6211 , 13.5431 , 13.5266 , 13.4637 , 13.4314 ,
13.376 , 13.4287 , 13.3966 , 13.3092 , 13.2671 , 13.1665 , 13.3573 , 13.2544 , 13.2116
, 13.113 , 13.3812 , 13.1991 , 13.1555 , 13.0582 , 13.3592 , 13.2887 , 13.2754 ,
13.2674};
        //Search for the input specie in the 'elements' array to start the caclculation
        for (i=0;i<211;i++){
                if(strcmp(specie, elements[i]) == 0 )
        {
                n=i;
            found = 1;
            break;
        }
        }
        //If the specie is found in the table
        if (found==1){
                //Use the atomic form factor formula which is the sum of the Gaussians
of a particular form
                result=a1[n]*exp(-b1[n]*pow(q/(4*M_PI),2))+a2[n]*exp(-
b2[n]*pow(q/(4*M_PI),2))+a3[n]*exp(-b3[n]*pow(q/(4*M_PI),2))+a4[n]*exp(-
b4[n]*pow(q/(4*M_PI),2))+c[n];
        }else{
                //Return error code in case the input specie is not found in the
database
                result=9898989898989898;
        }
        return result;
}
/*
The following function takes the value of h,k,l and atomic species array,
 as well as the corresponding x,y,z position arrays
and returns the real part of the structure factor for a gien value of h,k,l and theta
and lambda
*/
double realStructFactor(int h, int k, int l, double theta, double lambda, int nat, char
species[nat][10], double x[], double y[], double z[]){
        double result=0;
        int i;
        double q=4*M_PI*sin(theta*M_PI/180.0)/lambda;
        for (i=0;i<nat;i++){
                result=result+formFactorCalc(q,
species[i])*cos(2*M_PI*(h*x[i]+k*y[i]+l*z[i]));
        }
        return result;
}
/*
The following function takes the value of h,k,l and atomic species array,
```

```
 as well as the corresponding x,y,z position arrays
and returns the imaginary part of the structure factor for a gien value of h,k,l and
theta and lambda.
*/
double imagStructFactor(int h, int k, int l, double theta, double lambda, int nat, char
species[nat][10], double x[], double y[], double z[]){
        double result=0;
        int i;
        double q=4*M_PI*sin(theta*M_PI/180.0)/lambda;
        for (i=0;i<nat;i++){
                result=result+formFactorCalc(q,
species[i])*sin(2*M_PI*(h*x[i]+k*y[i]+l*z[i]));
        }
        return result;
}
/*Function to find out the no. of unique entries in an array,
 to get the unique entries of an array in a separate array,
 to count the no. of occurences of a given input in an array,
 to get the position of each unique entry in the given array*/
int uniqueCount(int size, double array[], double uniqueArray[], int pos[], int
count[]){
        int i,k,j;
        double temp;
        int size2;
        int found;
        for(i=0;i<size;i++){
                if(i==0){
                        j=0;
                        uniqueArray[j]=array[i];
                        pos[j]=i;
                        size2=1;
                        j++;
                }else{
                        found=0;
                        for(k=0;k<size2;k++){
                                if(array[i]==uniqueArray[k]){
                                        found=1;
                                        break;
                                }
                        }
                        if(found!=1){
                                uniqueArray[j]=array[i];
                                pos[j]=i;
                                j++;
                                size2++;
                        }
                }
        }
        for(k=0;k<size2;k++){
                int counter=0;
                for(i=0;i<size;i++){
                        if(uniqueArray[k]==array[i]){
                                counter++;
                        }
```

```
                }
                count[k]=counter;
        }
        return size2;
}
//Returns the non-zero entries in an array
int nonZeroEntries(int size, double array[]){
        int i;
        int count=0;
        for(i=0;i<size;i++){
                if(array[i]!=0){
                        count++;
                }
        }
        return count;
}

main(){
        int nat,h,k,l,i;
        double realSF, imagSF, F2;
        int h_arr[2000], k_arr[2000], l_arr[2000];
        double theta;
        double theta_arr[2000];
        double twoTheta_arr[2000];
        double lambda=1.54059;
        double dmin=lambda/2;
        int ibrav;
        double a,b,c;
    double alpha, beta, gamma;
    double dhkl;
    double dhkl_arr[2000];
    double Freal[2000], Fimag[2000],Fsq[2000];
    char input[30];
    printf("Enter Input File Name:n");
    scanf("%s",&input);
    char output1[30];
        strcpy(output1,input);
        char output2[30];
        strcpy(output2,input);
        FILE *fp=NULL;
        //INPUT FILE CONTAINING THE INFORMATION OF LATTICE TYPE, LATTICE PARAMS. AND
ATOMIC POSITIONS
        fp=fopen(strcat(input,".txt"),"r");
        //Read the first line that contains the number of atoms
        fscanf(fp,"%dn",&nat);
        //Read the second line that contains the bravais lattice type
        fscanf(fp,"%dn",&ibrav);
        //Arrays to store the atomic specie as well as the atomic positions
        double xpos[nat], ypos[nat], zpos[nat];
        char elem[nat][10];
        //Read the lattice parameters depending on the value of ibrav
    switch(ibrav){
        case 1: //Cubic
            //Read the lattice parameter
```

```
                fscanf(fp,"%lfn",&a);
                b=a;
                c=a;
                alpha=beta=gamma=90;
    break;
case 2: //Hexagonal
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&c);
    b=a;
    alpha=beta=90;
    gamma=120;
    break;
case 3: //Rhombohedral
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&alpha);
    b=a;
    c=a;
    beta=alpha;
    gamma=alpha;
    break;
case 4: //Tetragonal
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&c);
    b=a;
    alpha=beta=gamma=90;
    break;
case 5: //Orthorhombic
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&b);
                fscanf(fp,"%lfn",&c);
    alpha=beta=gamma=90;
    break;
case 6: //Monoclinic
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&b);
                fscanf(fp,"%lfn",&c);
                fscanf(fp,"%lfn",&beta);
    alpha=gamma=90;
    break;
case 7: //Triclinic
    //Read the lattice parameters
                fscanf(fp,"%lfn",&a);
                fscanf(fp,"%lfn",&b);
                fscanf(fp,"%lfn",&c);
                fscanf(fp,"%lfn",&alpha);
                fscanf(fp,"%lfn",&beta);
                fscanf(fp,"%lfn",&gamma);
    break;
default:
```

```
            //Read the lattice parameters
                        fscanf(fp,"%lfn",&a);
                        fscanf(fp,"%lfn",&b);
                        fscanf(fp,"%lfn",&c);
                        fscanf(fp,"%lfn",&alpha);
                        fscanf(fp,"%lfn",&beta);
                        fscanf(fp,"%lfn",&gamma);
            break;
    }
        //Skip the line containing the phrase "ATOMIC_POsition" the file
        fscanf(fp,"%*[^n]");
        //Start reading the atom symbol and the x,y,z coordinates
        for(i=0;i<nat;i++){
                fscanf(fp,"%st%lft%lft%lfn",&elem[i],&xpos[i],&ypos[i],&zpos[i]);
        }
        //RESULT OF XRD SIMULATOR
        //Store the reflection information in a file
        FILE *fp2=NULL;
        strcat(output1,"_reflections.txt");
        fp2=fopen(output1,"w");
        fprintf(fp2,"htktlt2thetatd_hkltFrealtFimagt|F|^2n");
        int j=0;
        for(h=-a/dmin;h<=a/dmin;h++){
        for(k=-b/dmin;k<=b/dmin;k++){
            for(l=-c/dmin;l<=c/dmin;l++){
                switch(ibrav){
                        case 1:
                                if(h==0&&k==0&&l==0){
                                        break;
                                                        }
                                                        //Get interplanar spacing using
the Cubic formula
                            dhkl=a/sqrt(h*h+k*k+l*l);
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                        }
                                                        //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                        break;
                                                         }else{
F2=realSF*realSF+imagSF*imagSF;
                                                                //Store inter-planar
spacing in an array
                                                               dhkl_arr[j]=dhkl;
                                                                //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
```

```
                                                    Fimag[j]=imagSF;
                                                            Fsq[j]=F2;
                                                            h_arr[j]=h;
                                                            k_arr[j]=k;
                                                            l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                             j++;
                                                            break;
                                                     }
                        case 2:
                            if(h==0&&k==0&&l==0){
                                        break;
                                                    }
                                                    //Get interplanar spacing using
the Hexagonal formula
                            dhkl=sqrt(pow(4.0/3.0*(h*h+h*k+k*k)/(a*a)+l*l/(c*c),-1));
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                    }
                                                    //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                    break;
                                                     }else{
F2=realSF*realSF+imagSF*imagSF;
                                                             //Store inter-planar
spacing in an array
                                                            dhkl_arr[j]=dhkl;
                                                             //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                    Fimag[j]=imagSF;
                                                            Fsq[j]=F2;
                                                            h_arr[j]=h;
                                                            k_arr[j]=k;
                                                            l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                             j++;
                                                            break;
                                                     }
                        case 3:
                            if(h==0&&k==0&&l==0){
                                        break;
                                                    }
                                                    //Get interplanar spacing using
the Rhombohedral formula
dhkl=sqrt(pow(((h*h+k*k+l*l)*sin(alpha*M_PI/180)*sin(alpha*M_PI/180)+2*(h*k+k*l+h*l)*(c
os(alpha*M_PI/180)*cos(alpha*M_PI/180)-
```

```
cos(alpha*M_PI/180)))/(a*a*(1-3*cos(alpha*M_PI/180)*cos(alpha*M_PI/180)+2*cos(alpha*M_P
I/180)*cos(alpha*M_PI/180)*cos(alpha*M_PI/180))),-1));
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                        }
                                                        //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                        break;
                                                         }else{
F2=realSF*realSF+imagSF*imagSF;
                                                                //Store inter-planar
spacing in an array
                                                               dhkl_arr[j]=dhkl;
                                                                //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                        Fimag[j]=imagSF;
                                                                Fsq[j]=F2;
                                                                h_arr[j]=h;
                                                                k_arr[j]=k;
                                                                l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                                 j++;
                                                                break;
                                                         }
                        case 4:
                            if(h==0&&k==0&&l==0){
                                        break;
                                                        }
                                                        //Get interplanar spacing using
the Tetragonal formula
                            dhkl=sqrt(pow((h*h+k*k)/a/a+l*l/c/c,-1));
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                        }
                                                        //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                        break;
                                                         }else{
F2=realSF*realSF+imagSF*imagSF;
                                                                //Store inter-planar
```

```
spacing in an array
                                                        dhkl_arr[j]=dhkl;
                                                         //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                Fimag[j]=imagSF;
                                                        Fsq[j]=F2;
                                                        h_arr[j]=h;
                                                        k_arr[j]=k;
                                                        l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                         j++;
                                                        break;
                                                         }
                            case 5:
                                if(h==0&&k==0&&l==0){
                                            break;
                                                }
                                                //Get interplanar spacing using
the Orthorhombic formula
                                dhkl=sqrt(pow(h*h/a/a+k*k/b/b+l*l/c/c,-1));
                                //if the corresponding angle is not possible
                                if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                    break;
                                                }
                                                //Get theta in radians
                                theta=asin(lambda/(2*dhkl));
                                //Convert theta to degrees
                                theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                break;
                                                 }else{
F2=realSF*realSF+imagSF*imagSF;
                                                         //Store inter-planar
spacing in an array
                                                        dhkl_arr[j]=dhkl;
                                                         //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                Fimag[j]=imagSF;
                                                        Fsq[j]=F2;
                                                        h_arr[j]=h;
                                                        k_arr[j]=k;
                                                        l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                         j++;
                                                        break;
                                                         }
                            case 6:
                                if(h==0&&k==0&&l==0){
```

```
                                    break;
                                                }
                                                //Get interplanar spacing using
the Monoclinic formula
dhkl=sqrt(pow((h*h/a/a+k*k*sin(beta*M_PI/180)*sin(beta*M_PI/180)/b/b+l*l/c/c-2*h*l*cos(
beta*M_PI/180)/a/c)/sin(beta*M_PI/180)/sin(beta*M_PI/180),-1));
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                }
                                                //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                break;
                                                 }else{
F2=realSF*realSF+imagSF*imagSF;
                                                        //Store inter-planar
spacing in an array
                                                       dhkl_arr[j]=dhkl;
                                                        //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                Fimag[j]=imagSF;
                                                        Fsq[j]=F2;
                                                        h_arr[j]=h;
                                                        k_arr[j]=k;
                                                        l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                         j++;
                                                        break;
                                                 }
                        case 7:
                            if(h==0&&k==0&&l==0){
                                    break;
                                                }
                                                //Get interplanar spacing using
the Monoclinic formula
dhkl=sqrt(pow((h*h/a/a*pow(sin(alpha*M_PI/180),2)+k*k/b/b*pow(sin(beta*M_PI/180),2)+l*l
/c/c*pow(sin(gamma*M_PI/180),2)+2*k*l*cos(alpha*M_PI/180)/b/c+2*h*l*cos(beta*M_PI/180)/
a/c+2*h*k*cos(gamma*M_PI/180)/b/a)/(1-pow(cos(alpha*M_PI/180),2)-
pow(cos(beta*M_PI/180),2)-
pow(cos(gamma*M_PI/180),2)+2*cos(alpha*M_PI/180)*cos(beta*M_PI/180)*cos(gamma*M_PI/180)
),-1));                             //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                }
                                                //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
```

```
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                    break;
                                                     }else{
F2=realSF*realSF+imagSF*imagSF;
                                                            //Store inter-planar
spacing in an array
                                                    dhkl_arr[j]=dhkl;
                                                     //Store theta in array
                                theta_arr[j]=theta;
                                twoTheta_arr[j]=theta*2;
                                Freal[j]=realSF;
                                                    Fimag[j]=imagSF;
                                                            Fsq[j]=F2;
                                                            h_arr[j]=h;
                                                            k_arr[j]=k;
                                                            l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                             j++;
                                                            break;
                                                     }
                                default:
                                    if(h==0&&k==0&&l==0){
                                                break;
                                                    }
                                                    //Get interplanar spacing using
the Triclinic formula
dhkl=sqrt(pow((h*h/a/a*pow(sin(alpha*M_PI/180),2)+k*k/b/b*pow(sin(beta*M_PI/180),2)+l*l
/c/c*pow(sin(gamma*M_PI/180),2)+2*k*l*cos(alpha*M_PI/180)/b/c+2*h*l*cos(beta*M_PI/180)/
a/c+2*h*k*cos(gamma*M_PI/180)/b/a)/(1-pow(cos(alpha*M_PI/180),2)-
pow(cos(beta*M_PI/180),2)-
pow(cos(gamma*M_PI/180),2)+2*cos(alpha*M_PI/180)*cos(beta*M_PI/180)*cos(gamma*M_PI/180)
),-1));
                            //if the corresponding angle is not possible
                            if((lambda/(2*dhkl))>1||(lambda/(2*dhkl))<(-1)){
                                break;
                                                    }
                                                    //Get theta in radians
                            theta=asin(lambda/(2*dhkl));
                            //Convert theta to degrees
                            theta=theta*180/M_PI;
realSF=realStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
imagSF=imagStructFactor(h,k,l,theta,lambda,nat,elem,xpos,ypos,zpos);
if(fabs(realSF-0)<=0.01&&fabs(imagSF-0)<=0.01){
                                                    break;
                                                     }else{
F2=realSF*realSF+imagSF*imagSF;
                                                            //Store inter-planar
spacing in an array
                                                    dhkl_arr[j]=dhkl;
                                                     //Store theta in array
                                theta_arr[j]=theta;
```

```
                                    twoTheta_arr[j]=theta*2;
                                    Freal[j]=realSF;
                                                                Fimag[j]=imagSF;
                                                                        Fsq[j]=F2;
                                                                        h_arr[j]=h;
                                                                        k_arr[j]=k;
                                                                        l_arr[j]=l;
fprintf(fp2,"%dt%dt%dt%lft%lft%lft%lft%lfn",h,k,l,2*theta,dhkl,realSF,imagSF,F2);
                                                                         j++;
                                                                        break;
                                                                 }
                                }
                            }
                        }
                    }
                    //A lot of extra redundant kind of stuff that probably wasn't needed
                    //Basically  this block of code is suposed to give the final information that will
be plotted
                    int totalNoOfReflections=nonZeroEntries(2000,theta_arr);
                    double hUnique[2000];
                    double kUnique[2000];
                    double lUnique[2000];
                    double dhklUnique[2000];
                        double thetaUnique[2000];
                        int multiplicity[2000];
                        int pos[2000];
                        int
nUnique=uniqueCount(totalNoOfReflections,theta_arr,thetaUnique,pos,multiplicity);
                        for(i=0;i<nUnique;i++){
                                printf("%lft%dt%dn",2*thetaUnique[i],pos[i],multiplicity[i]);
                        }
                        //Lorentz-Polarization Correction
                        double intensity[nUnique];
                        for(i=0;i<nUnique;i++){
                                intensity[i]=multiplicity[i]*Fsq[pos[i]];
intensity[i]=intensity[i]*(1+cos(twoTheta_arr[pos[i]]*M_PI/180)*cos(twoTheta_arr[pos[i]
]*M_PI/180));
intensity[i]=intensity[i]/(sin(theta_arr[pos[i]]*M_PI/180)*sin(theta_arr[pos[i]]*M_PI/1
80)*cos(theta_arr[pos[i]]*M_PI/180));
                        }
                        //Final plottable results go in the file given by fp3
                        FILE *fp3=NULL;
                        strcat(output2,"_plotXRD.txt");
                        fp3=fopen(output2,"w");
                        fprintf(fp3,"htktlt2thetatd_hklt|F|^2tIntensitytMultiplicityn");
                        for(i=0;i<nUnique;i++){
fprintf(fp3,"%dt%dt%dt%lft%lft%lft%lft%dn",h_arr[pos[i]],k_arr[pos[i]],l_arr[pos[i]],2*
theta_arr[pos[i]],dhkl_arr[pos[i]],Fsq[pos[i]],intensity[i],multiplicity[i]);
                        }
}
```

Input File Examples:

Fe_BCC.txt

```
2
1
2.848
ATOMIC_POSITIONS {crystal}
Fe 0.00 0.00 0.00
Fe 0.50 0.50 0.50
```

Cu_FCC.txt

```
4
1
3.6149
ATOMIC_POSITIONS {crystal}
Cu 0.000000 0.000000 0.000000
Cu 0.000000 0.500000 0.500000
Cu 0.500000 0.000000 0.500000
Cu 0.500000 0.500000 0.000000
```

ZnO_Hex.txt

```
4
2
3.2533
5.2073
Atomic Position
Zn 0.333330 0.666670 0.000000
Zn 0.666670 0.333340 0.500000
O 0.333330 0.666670 0.382000
O 0.666670 0.333340 0.882000
```

CdS_Cub.txt

```
8
1
5.94083
ATOMIC_POSITIONS {crystal}
Cd 0.000000 0.000000 0.000000
Cd 0.000000 0.500000 0.500000
Cd 0.500000 0.000000 0.500000
Cd 0.500000 0.500000 0.000000
S 0.250000 0.250000 0.750001
S 0.250000 0.750001 0.250000
S 0.750001 0.250000 0.250000
S 0.750001 0.750001 0.750001
```

TiO2_Tetra.txt

```
6
4
4.65178
2.96991
ATOMIC_POSITIONS {crystal}
Ti 0.500000 0.500000 0.500001
```

```
Ti 0.000000 0.000000 0.000000
O 0.695090 0.695090 0.000000
O 0.195089 0.804912 0.500001
O 0.304911 0.304911 0.000000
O 0.804912 0.195089 0.500001
```

OUTPUT:

```
manas@ubuntu:~/XRD Simulator$ ./a.out
Enter Input File Name:
Fe_BCC
117.585186      0       24
139.084846      4       8
99.814188       5       12
82.983229       7       24
65.494963       10      6
44.977321       21      12
manas@ubuntu:~/XRD Simulator$
```

Output Files Generated:

`Fe_BCC_plotXRD.txt` and `Fe_BCC_reflections.txt`

Now, the file with the suffix `plotXRD.txt` contains the plottable data, i.e. the intensity as well as the 2theta values. 2theta values in the 4th column and the intensity in the 7th column. You can plot these using gnuplot using the impulse chart type. Or using Origin using stem scatter chart.

To make things easier for you guys, I am also attaching a shell script down below, that can be used to create a Gnuplot script that would plot a very neat looking graph even with the peaks labelled using the hkl miller indices.

GNUPLOT Script:

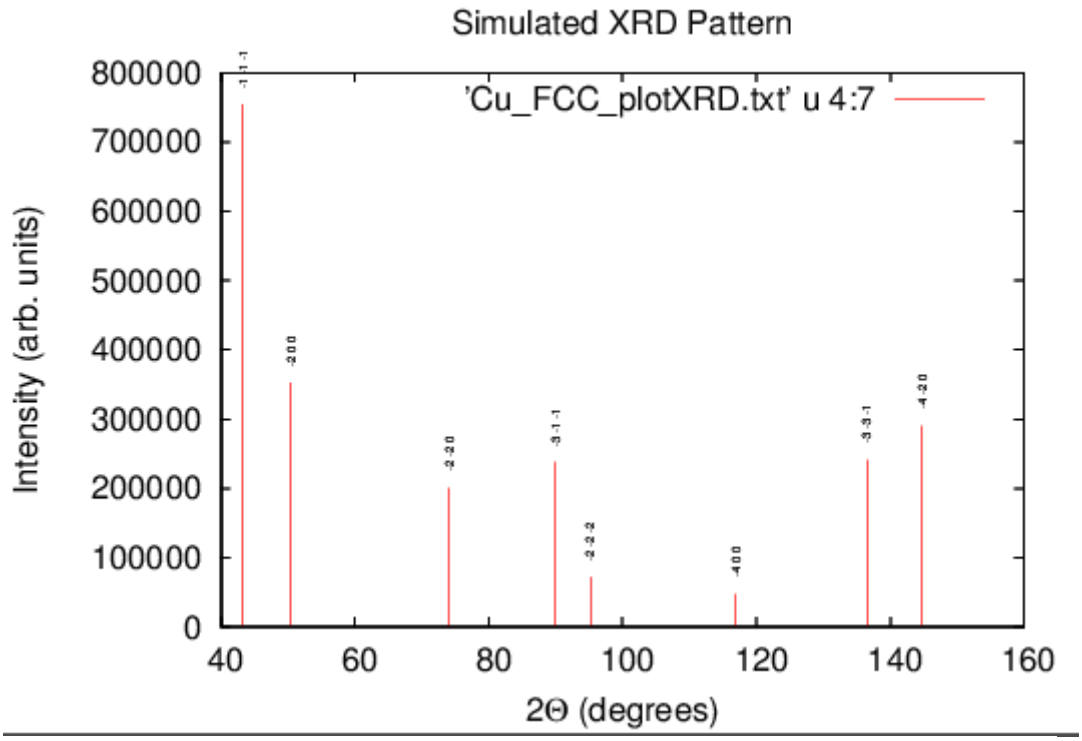
To make the following script work, save it as `xrdPlotter.sh` and then in your terminal run,
`chmod u+x xrdPlotter.sh` to make it executable
and then run the script using `./xrdPlotter.sh Fe_BCC`

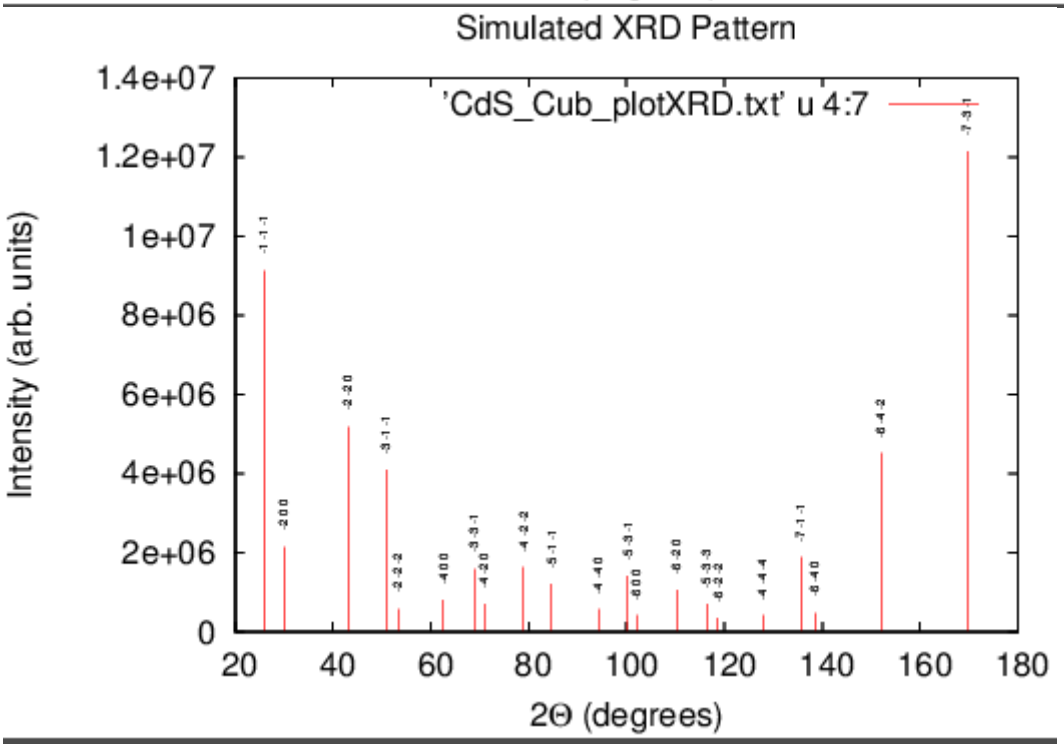
```
#!/bin/bash
filename=$1
filename=$(echo $1'_plotXRD.txt')
n=$(wc -l <$filename)
echo "
set terminal png size 1000,500
set output '"$1"plot.png'
set xlabel '2{/Symbol Q} (degrees)'
set ylabel 'Intensity (arb. units)'
set title 'Simulated XRD Pattern'" >XRDplotScript.p

for (( i=2; i<=$n; i++ ))
do
        h=$(awk 'NR=='$i'{print $1}' $filename)
        k=$(awk 'NR=='$i'{print $2}' $filename)
        l=$(awk 'NR=='$i'{print $3}' $filename)
        index=$(echo $h $k $l)
```

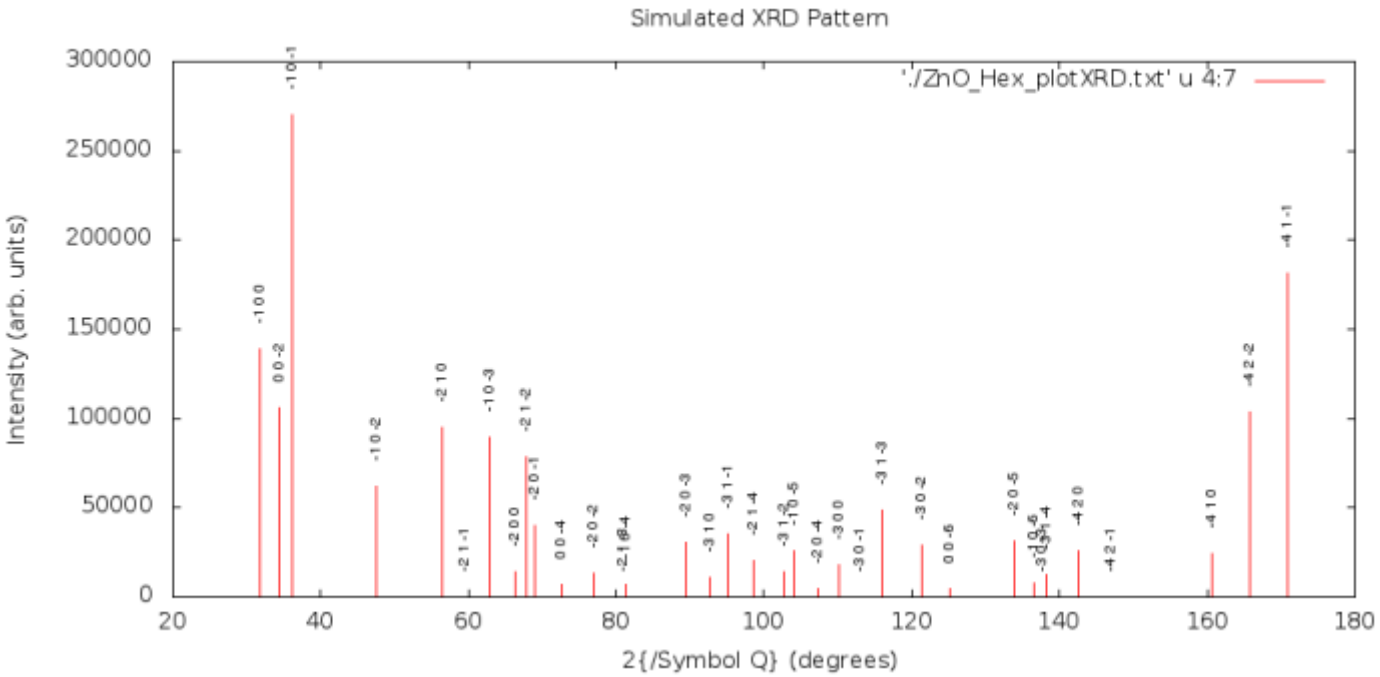
```
        x=$(awk 'NR=='$i'{print $4}' $filename)
        y=$(awk 'NR=='$i'{print $7}' $filename)
        echo "
set label '"$index"' at "$x","$y" left rotate by 90 offset 0,0.5 font 'Helvetica,8'"
>>XRDplotScript.p
done


echo "
#set key box linestyle 1
plot '"$filename"' u 4:7 w impulse">>XRDplotScript.p

echo "
set terminal postscript enhanced color solid 22
set output '"$1"plot.eps'
set xlabel '2{/Symbol Q} (degrees)'
set ylabel 'Intensity (arb. units)'
set title 'Simulated XRD Pattern'" >>XRDplotScript.p

for (( i=2; i<=$n; i++ ))
do
        h=$(awk 'NR=='$i'{print $1}' $filename)
        k=$(awk 'NR=='$i'{print $2}' $filename)
        l=$(awk 'NR=='$i'{print $3}' $filename)
        index=$(echo $h $k $l)
        x=$(awk 'NR=='$i'{print $4}' $filename)
        y=$(awk 'NR=='$i'{print $7}' $filename)
        echo "
set label '"$index"' at "$x","$y" left rotate by 90 offset 0,0.5 font 'Helvetica,8'"
>>XRDplotScript.p
done


echo "
#set key box linestyle 1
plot '"$filename"' u 4:7 w impulse">>XRDplotScript.p


gnuplot ./XRDplotScript.p
```

Gnuplot Output:

On the execution of the above scripts there will two plots generated in `.esp` and `.png` format, that look like the following:


Simulated XRD Pattern
'Cu_FCC_plotXRD.txt' u 4:7
Intensity (arb. units)
2Θ (degrees)
800000
700000
600000
500000
400000
300000
200000
100000
0
40
60
80
100
120
140
160


Simulated XRD Pattern
'CdS_Cub_plotXRD.txt' u 4:7
Intensity (arb. units)
2Θ (degrees)
1.4e+07
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
20
40
60
80
100
120
140
160
180



References:

http://pd.chem.ucl.ac.uk/pdnn/chapter.htm
http://pd.chem.ucl.ac.uk/pdnn/powintro/introind.htm
http://pd.chem.ucl.ac.uk/pdnn/diff2/dindex2.htm
https://www.bragitoff.com/2017/08/x-ray-diffraction-xrd/

Manas Sharma
I'm a physicist specializing in theoretical, computational and experimental condensed matter physics. I like to develop Physics related apps and softwares from time to time. Can code in most of the popular languages. Like to share my knowledge in Physics and applications using this Blog and a YouTube channel.

Share this:

Like this:

Like Loading...