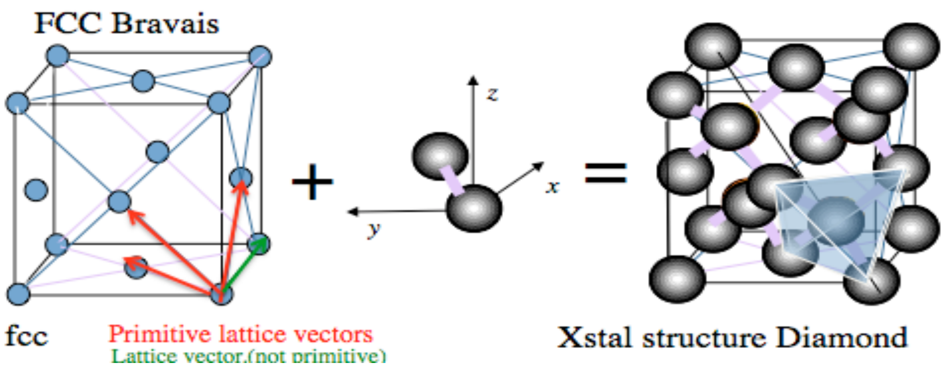
```
> restart;
> with(geom3d):with(Spread):
```

PROBLEM 200-2 solved

We will generate all the points for the atom locations, $\boldsymbol{X}$
by first generating the Bravais lattice points, $\boldsymbol{R} = n_1\boldsymbol{a}_1+n_2\boldsymbol{a}_2+n_3\boldsymbol{a}_3;\ n_1,n_2,n_3$ *integer*
$\boldsymbol{a}_1$= [1/2, 0 ,1/2]; vector in a cartesian coordinate system
$\boldsymbol{a}_2$= [0, 1/2, 1/2];
$\boldsymbol{a}_3$= [1/2, 1/2, 0];
,then for each Bravais lattice point we have two atom locations given by $\boldsymbol{T} = n_4\boldsymbol{t},\ n_4 \in$ [0,1]
$\boldsymbol{t}$= [1/4, 1/4, 1/4];
$\boldsymbol{X} = \boldsymbol{R} + \boldsymbol{T} = [1/2\cdot(n_1+n_3)+1/4\cdot n_4,\ 1/2\cdot(n_2+n_3)+1/4\cdot n_4,\ 1/2\cdot(n_1+n_2)+1/4\cdot n_4\ ];$
The strategy is just generating a lot of unique lattice points around origo by translating the primitive unit cell, generating the Bravais lattice, and assign the two atom positions for each Bravais lattice point, and just measure the distance to the origo and put the result in a table for later processing/sorting

```
> point(origo,0,0,0):thelist:='thelist':
  atnum:=0:
  for n1 from -8 to 7 do
   for n2 from -8 to 7 do
    for n3 from -8 to 7 do
      #done for each primitive cell;
      for n4 from 0 to 1 do # two atoms pr primitive cell
        atnum:=atnum+1;
        point(coords,[1/2*(n1+n3)+1/4*n4,
                      1/2*(n2+n3)+1/4*n4,
                      1/2*(n1+n2)+1/4*n4]);
        d:=distance(coords,origo);
        thelist[atnum]:=evalf(d);
        #print(atnum);
       od;#n4
     od;#n3
     od;#n2
    od;#n1
   maxatnum:=atnum;
```

$$maxatnum := 8192 \quad (1)$$



Sort the list (of distances from the origo)

```
> sortedlist:=sort([seq(thelist[j],j=1..maxatnum)]):
```

We count how many numbers in the list are equal, this gives the number of atoms in each shell; NumInShell. We test for equality by checking that the difference between the previous number in the list is larger than a constant chosen to be larger than any inaccuries or rounding errors in calculations, = delta.   We put the results in a spread sheed , ss, in the following.

```
> ss := CreateSpreadsheet(Shells): SetCellFormula(ss, 1, 1,
  'ShellNo'):SetCellFormula(ss, 1, 2, 'Num_in_Shell'):
  for j from 3 to 5 do
     SetCellFormula(ss, 1, j, 'ShellDistance');
  od:
  ShellNum:=0:d:=0:NumInShell:=0:delta:=0.01:
  for j from 1 to maxatnum do
   if ((sortedlist[j]-d)>=delta)
    then
     if ShellNum<36#No need to go beyound shell 36
        then
           x:=sqrt(convert(d^2,rational));
           if ShellNum>0
              then
                 RowNum:=ShellNum+1;
                 SetCellFormula(ss, RowNum,1 , ShellNum);
                 SetCellFormula(ss, RowNum,2, NumInShell);
                 SetCellFormula(ss, RowNum,3 , x*a);
                 SetCellFormula(ss, RowNum,4 , d*a);
                 SetCellFormula(ss, RowNum,5 , d*0.43596*nm);
             end if;
       end if;
     ShellNum:=ShellNum+1;d:=sortedlist[j];
     NumInShell:=0:
   end if:
   NumInShell:=NumInShell+1:
  od:
```

Shells

	A	B	C	D	E	F
1	*ShellNo*	*Num_in_Shell*	*ShellDistance*	*ShellDistance*	*ShellDistance*	
2	1	4	$\frac{1}{4}\sqrt{3}\,a$	$0.4330\,a$	$0.1888\,nm$	
3	2	12	$\frac{1}{2}\sqrt{2}\,a$	$0.7071\,a$	$0.3083\,nm$	
4	3	12	$\frac{1}{4}\sqrt{11}\,a$	$0.8292\,a$	$0.3615\,nm$	
5	4	6	a	$1.0000\,a$	$0.4360\,nm$	
6	5	12	$\frac{1}{4}\sqrt{19}\,a$	$1.0897\,a$	$0.4751\,nm$	
7	6	24	$\frac{1}{2}\sqrt{6}\,a$	$1.2247\,a$	$0.5339\,nm$	
8	7	16	$\frac{3}{4}\sqrt{3}\,a$	$1.2990\,a$	$0.5663\,nm$	

(2)