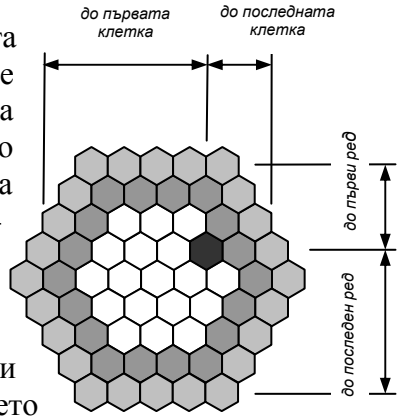
АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ХЕКСАГОН

Да наречем *рамка* клетките с еднакво минимално разстояние до страните на полето. Очевидно цялото поле се състои от $n - 1$ концентрично разположени рамки. Всяка по-вътрешна рамка има 6 клетки по-малко от съседната ѝ по-външна, т.е. количествата клетки в концентричните рамки образуват аритметична прогресия с разлика 6.



В подзадача А определяме броя рамки f от клетката (p, q) до страните на полето. Видно от схемата е, че f е минимума от разстоянието от клетка (p, q) до първата клетка в реда, до последната клетка в реда, до първия ред, до последния ред в полето. Ще използваме свойствата на аритметичната прогресия за бързо изчисляване на *cells* – общия брой клетки в f-те рамки [ред 31 от сорс-кода]. За рамката, в която се намира, клетката (p, q) има някакъв пореден номер. Определяме го като се съобразяваме с броя клетки в страната на рамката [редове 32-39 от сорс-кода] и го добавяме към *cells*. Така получаваме колко клетки от полето бихме обходили според правилата на задачата, за да достигнем до клетка (p, q). Ако $cells > m$, клетката е празна. В противен случай в клетката е записано *cells*-то поредно число от зададените и стойността му е $cells + s - 1$. Заради ограниченията в задачата отговорът може да е извън обхвата на 64 битовите цели числа и това налага да приложим някаква реализация за събиране на дълги числа [редове 10-20 от сорс-кода].

В подзадача Б с двоично търсене определяме в коя рамка е клетката с m-то написано число (то е най-голямото в полето) [редове 47-53 от сорс-кода]. Определяме поредността i на обхождане на клетката в границите на рамката [ред 55 от сорс-кода]. Като се съобразяваме с броя клетки в страната на рамката, трансформираме i в координати (ред, позиция) на клетката спрямо рамката [редове 59-66 от сорс-кода]. С отчитане на външните за клетката рамки преобразуваме намерените ѝ координати в координати спрямо цялото поле.

Автор: Евгений Василев

Сорс-кодът на програмата

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  typedef unsigned long long ull;
6  ull s, p, q, n, m;
7
8  ull cn(ull n, ull f){ return (n*2 -1 -f)*f*3;}
9
10 void print_sum_of(ull a, ull b){
11   ull c=1000000000, al=a%c+b%c;
12   a=a/c+b/c+al/c;
13   al%=c;
14   if (a) {
15     char fillchar=cout.fill('0');
```

```
    cout<<a<<setw(9)<<al<<endl;
    cout.fill(fillchar);
  }
  else cout<<al<<endl;
}

void taskA(ull n, ull row, ull pos){
  ull frames=row-1, cells=0, i;
  i = n*2 -1 -row;                              // To down
  if (i<frames) frames = i;
  i = pos-1;                                    // To left
  if (i<frames) frames = i;
  i=(row<=n?(n-1+row-pos):(3*n-1-row-pos));     // To right
  if (i<frames) frames = i;

  cells=cn(n,frames);
  row-=frames;
  pos-=frames;
  n-=frames;

  if (row==1) cells += pos;
  else if (row==(n*2 -1)) cells += n*4 - 2 -pos;
  else if (pos>1) cells += n - 1 + row;
  else cells += 6*n-4-row;

  if (cells>m) cout<<"0\n";
  else print_sum_of(cells-1,s);
}


void taskB(ull n, ull m){
  ull hi_frames=n,lo_frames=0,c_frame,i,j;
  do {
    c_frame = (hi_frames+lo_frames)/2;
    i=cn(n,c_frame);
    if (i<m)lo_frames=c_frame+1 ;
    else hi_frames=c_frame;
  } while (lo_frames<hi_frames);

  i=m-cn(n,lo_frames-1);
  n-=lo_frames-1;
  j=--i%(--n);

  switch (i/n) {
    case 0: i=0;break;
    case 1: i=j;j+=n;break;
    case 2: i=n+j;j=2*n-j;break;
    case 3: i=2*n,j=n-j;break;
    case 4: n*=2;
    default:i=n-j;j=0;
  }
  cout<<lo_frames+i<<" "<<lo_frames+j<<endl;
}

int main(){

  cin>>n>>m>>s>>p>>q;
  taskA(n,p,q);
  taskB(n,m);

  return 0;
}
```