1. #include <bits/stdc++.h>
2. #include <math.h>
3. #include <fstream>
4. **using** **namespace** std;
5. #define IOS ios::sync\_with\_stdio();cin.tie(0);
6. #define endl '\n'
7. #define M\_PI 3.14159265358979323846 // pi
8. #define ISC 1.382 //Solar Constant(=1.382 [kW /m)
9. #define sun 149600000000 // 太陽到地球的距離
10. #define F first
11. #define S second
12. #define pii pair<int,int>
13. **long** **double** all = 0;
14. **int** n;
15. **int** s[10005][10005]={};
16. pii ra,rb;
17. **bool** is\_solar[10005][10005]={};
18. **inline** **long** **double** f\_fi(**double** **long** del,**double** **long** om,**long** **double** cos\_sis,**long** **double** phi){
19. **long** **double** cos\_fi = (sin(del) \* cos(phi) - cos(om) \* cos(del) \* sin(phi)) / (sin(acos(cos\_sis)));
20. **long** **double** fi = acos(cos\_fi)\*180/M\_PI;
21. **if**(om > 0){
22. fi = 180 - fi;
23. }**else** **if**(om <0){
24. fi -= 180;
25. }**else**{
26. fi = 0;
27. }
28. fi = fi\*M\_PI/180;
29. **return** cos\_fi;
30. }
31. **inline** **long** **double** get\_IBR(**long** **double** N,**long** **double** R,**long** **double** B,**long** **double** phi,**long** **double** lam,**long** **double** om, **long** **double**\* s\_sin\_H,**long** **double**\* s\_fi,**long** **double**\* s\_IsBR,**long** **double** \*s\_IrBR){
32. om = ((om) \* M\_PI)/180;
33. **long** **double** X = [](**long** **double** N){**return** (2 \* M\_PI \* (N - 1))/365;}(N);
34. **long** **double** del = [](**long** **double** X,**long** **double** N){**return** 0.006918 - 0.399912 \* cos(X) + 0.070257 \* sin(X) - 0.006758 \* cos(2 \* X) + 0.000908 \* sin(2 \* X);}(X,N);
35. **long** **double** sin\_H = [](**long** **double** cos\_phi,**long** **double** cos\_del,**long** **double** cos\_om,**long** **double** del,**long** **double** phi){**return** (cos\_phi \* cos\_del \* cos\_om) + (sin(del) \* sin(phi));}(cos(phi),cos(del),cos(om),del,phi);
36. **long** **double** sin\_H0 = [](**long** **double** del,**long** **double** phi){**return** cos(del - phi);}(del,phi);
37. **long** **double** I = [](**long** **double** sin\_H,**long** **double** sin\_H0){**return** (0.42 \* sin\_H) + (((2.92 - sin\_H0) / (2 \* sin\_H0)) \* pow(sin\_H,2)) - (((2.92 - sin\_H0) / (4 \* pow(sin\_H0,2))) \* pow(sin\_H,3));}(del,phi);
38. **long** **double** IDN = [](**long** **double** I,**long** **double** sin\_H){**return** ((1.323 \* I) / (sin\_H)) - 0.5466;}(I,sin\_H);
39. **long** **double** cos\_si = [](**long** **double** del,**long** **double** om,**long** **double** B,**long** **double** R,**long** **double** phi){**return** (sin(phi) \* cos(B) - cos(phi) \* sin(B) \* cos(R)) \* sin(del) + (cos(phi) \* cos(B) + sin(phi) \* sin(B) \* cos(R)) \* cos(del) \* cos(om) + sin(B) \* sin(R) \* cos(del) \*sin(om);}(del,om,B,R,phi);
40. **long** **double** IoH = [](**long** **double** sin\_H,**long** **double** N){**return** ISC \* (1 + 0.033 \* cos((N - 2) \* ((2 \* M\_PI) / (365)))) \* sin\_H;}(sin\_H,N);
41. **long** **double** ISH = [](**long** **double** I,**long** **double** IDN,**long** **double** sin\_H){**return** I - (IDN \* sin\_H);}(I,IDN,sin\_H);
42. **long** **double** ro = 0.55;
43. **long** **double** IbBR = [](**long** **double** IDN,**long** **double** cos\_si){**return** IDN \* cos\_si;}(IDN,cos\_si);
44. **long** **double** IsBR = [](**long** **double** I,**long** **double** ISH,**long** **double** IoH,**long** **double** cos\_si,**long** **double** sin\_H,**long** **double** B){**return** ISH \* (((I-ISH) / (IoH)) \* ((cos\_si) / (sin\_H)) + (1 + ((I-ISH) / (IoH))) \* ((1 + cos(B)) / (2)));}(I,ISH,IoH,cos\_si,sin\_H,B);
45. **long** **double** IrBR = [](**long** **double** I,**long** **double** ro,**long** **double** B){**return** (ro \* I) \* ((1 - cos(B)) / (2));}(I,ro,B);
46. **long** **double** fi = f\_fi(del,om,sin\_H,phi);
47. \*s\_sin\_H = sin\_H;
48. \*s\_fi = fi;
49. \*s\_IsBR = IsBR;
50. \*s\_IrBR = IrBR;
51. all += IbBR;
52. **return** IsBR+IbBR+IrBR;
53. }
54. **inline** **bool** init(**long** **double**\* phi,**long** **double**\* lam){
55. **int** m = 1;
56. **char** address[500];
57. cout << "input the file name:";
58. cin >> address;
59. cout << "input Latitude:";
60. cin >> \*phi;
61. \*phi = ((\*phi) \* M\_PI)/180;
62. cout << "input Longitude:";
63. cin >> \*lam;
64. \*lam = ((\*lam) \* M\_PI)/180;
65. **for**(**int** i = 0; i < m; i++){
66. cout << "input the solar upper left corner:";
67. cin >> ra.F >> ra.S;
68. cout << "input the solar lower right corner:";
69. cin >> rb.F >> rb.S;
70. **for**(**int** i = ra.F; i <= rb.F; i++){
71. **for**(**int** q= ra.S; q <= rb.S; q++){
72. is\_solar[i][q] = 1;
73. }
74. }
75. }
76. fstream file;
77. file.open(address,ios::in);
78. **if**(!file){
79. cout << "can't find file!"<<endl;
80. system("pause");
81. **return** 0;
82. }
83. file >> n;
84. **for** (**int** i = 0; i < n; i++){
85. **for**(**int** q = 0; q < n; q++){
86. file >> s[i][q];
87. }
88. }
89. **for**(**int** i = 0; i < n; i++){
90. **for**(**int** q = 0; q < n; q++){
91. cout << s[i][q] << ' ' ;
92. }
93. cout << endl;
94. }
95. **return** 1;
96. }
97. **void** get\_hourse(queue<pii>& qu,pii now,**long** **double** sin\_H,**long** **double** fi){
98. **long** **double** X = ((sun \* cos(asin(sin\_H))) \* sin(fi) \* -1),Y = ((sun \* cos(asin(sin\_H))) \* cos(fi));
99. **int** px,py;
100. **if**(X > now.F)px = 1;
101. **else** px = -1;
102. **if**(Y > now.S)py = 1;
103. **else** py = -1;
104. **if**(X - now.F == 0){
105. **for**(**int** i = now.S; i >=0 && i < n; i += py){
106. qu.push(pii(i,now.F));
107. }
108. }**else** **if**(Y - now.S == 0){
109. **for**(**int** i = now.F; i >=0 && i < n; i += px){
110. qu.push(pii(now.S,i));
111. }
112. }**else**{
113. **long** **double** m = (Y - now.S) / (X - now.F);
114. **int** nx = now.F,ny = now.S;
115. **while**(nx >= 0 && nx < n&& ny >=0 && ny < n){
116. **while**(ny < n && ny >= 0){
117. **if**(abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) > 0.5){
118. ny += py;
119. }**else**{
120. **break**;
121. }
122. }
123. **for**(**int** i = ny; i >= 0 && i < n ; i += py){
124. **if**(abs(i - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) <= 0.5 && (nx != now.F || i !=now.S)){
125. qu.push(pii(i,nx));
126. }**else** **if**(abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) > 0.5){
127. **break**;
128. }
129. }
130. nx += px;
131. }
132. }
133. **return**;
134. }
135. pii is\_out(**long** **double** sin\_H,**long** **double** fi){
136. **int** re = 0,cont = 0;
137. **for**(**int** i = ra.F; i <= rb.F; i++){
138. **for**(**int** q = ra.S; q <= rb.S; q++){
139. **if**(is\_solar[i][q]){
140. cont++;
141. queue<pii> qu;
142. get\_hourse(qu,pii(q,i),sin\_H,fi);
143. **long** **double** lb = asin (sin\_H) \* 180.0 / M\_PI;
144. **while**(!qu.empty()){
145. pii now = qu.front();
146. **long** **double** lx = sqrt( (now.F - i) \* (now.F - i) + (now.S - q) \* (now.S - q) );
147. **if**((s[now.F][now.S] < s[i][q]) || ( lb > atan((s[now.F][now.S] - s[i][q]) / lx) \* 180.0 / M\_PI)){
148. qu.pop();
149. }**else**{
150. **break**;
151. }
152. }
153. **if**(qu.size() == 0){
154. re++;
155. }
156. }
157. }
158. }
159. **return** pii(re,cont);
160. }
161. **void** solve(**long** **double** phi, **long** **double** lam,**long** **double** ans[]){
162. **int** N[5] = {113,174,267,357};
163. **for**(**int** B = 0; B <= 90; B++){
164. **for**(auto q:N){
165. **for**(**int** i = -75; i <= 75; i += 15){
166. **long** **double** sin\_H,fi,IsBR,IrBR;
167. **long** **double** IBR = get\_IBR(N[q],((0) \* M\_PI)/180,((B) \* M\_PI)/180,phi,lam,i,&sin\_H,&fi,&IsBR,&IrBR);
168. pii tmp = is\_out(sin\_H,fi);
169. ans[B] += IBR \* tmp.F;
170. ans[B] += IsBR \* (tmp.S - tmp.F);
171. }
172. }
173. }
174. **for**(**int** i = 0; i <= 90; i++){
175. cout << ans[i] << ',';
176. }
177. }
178. **void** output(**long** **double** ans[95]){
179. fstream file;
180. file.open("out.csv",ios::out);
181. **for**(**int** i = 0; i <= 90; i++){
182. file << ans[i];
183. file << ',';
184. }
185. file.close();
186. }
187. **int** main(){
188. IOS;**long** **double** phi,lam = -5;
189. **long** **double** ans[95] = {};
190. **if**(!init(&phi,&lam))**return** 0;
191. solve(phi,lam,ans);
192. output(ans);
193. system("pause");
194. **return** 0;
195. }