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. ///////////////////////////////debug////////////////////////////////////////
14. //#define grorge
15. #ifdef grorge
16. #define debug(...) do{\
17. fprintf(stderr,"%s - %d (%s) = ",\_\_PRETTY\_FUNCTION\_\_,\_\_LINE\_\_,#\_\_VA\_ARGS\_\_);\
18. \_do(\_\_VA\_ARGS\_\_);\
19. }**while**(0)
20. **template**<**typename** T>**void** \_do(T &&\_x){cerr<<\_x<<endl;}
21. **template**<**typename** T,**typename** ...S> **void** \_do(T &&\_x,S &&...\_t){cerr<<\_x<<" ,";\_do(\_t...);}
22. **template**<**typename** \_a,**typename** \_b> ostream& operator << (ostream &\_s,**const** pair<\_a,\_b> &\_p){**return** \_s<<"("<<\_p.X<<","<<\_p.Y<<")";}
23. **template**<**typename** It> ostream& \_OUTC(ostream &\_s,It \_ita,It \_itb)
24. {
25. \_s<<"{";
26. **for**(It \_it=\_ita;\_it!=\_itb;\_it++)
27. {
28. \_s<<(\_it==\_ita?"":",")<<\*\_it;
29. }
30. \_s<<"}";
31. **return** \_s;
32. }
33. **template**<**typename** \_a> ostream &operator << (ostream &\_s,vector<\_a> &\_c){**return** \_OUTC(\_s,ALL(\_c));}
34. **template**<**typename** \_a> ostream &operator << (ostream &\_s,set<\_a> &\_c){**return** \_OUTC(\_s,ALL(\_c));}
35. **template**<**typename** \_a,**typename** \_b> ostream &operator << (ostream &\_s,map<\_a,\_b> &\_c){**return** \_OUTC(\_s,ALL(\_c));}
36. **template**<**typename** \_t> **void** pary(\_t \_a,\_t \_b){\_OUTC(cerr,\_a,\_b);cerr<<endl;}
37. #else
38. #define debug(...)
39. #define pary(...)
40. #endif
41. ///////////////////////////////////////////////////////////////////////////////
42. //ψ = fi
43. //δ = del
44. //ω = om
45. //φ = phi 緯度
46. //e = E
47. //χ = X
48. //n = N
49. //λ = lam 經度
50. //ρ = ro
51. //β = B 太陽能板的傾斜角
52. //γ = R 太陽能板的方位角
53. //θ = si
54. **struct** cycle{
55. //球座標 ( r , θ , φ )
56. //r = 半徑 θ = 跟z軸夾角 φ = 跟x軸夾角
57. };
58. **long** **double** all = 0;
59. **int** n;
60. **int** s[10005][10005]={};
61. pii ra,rb;
62. **bool** is\_solar[10005][10005]={};
63. **inline** **long** **double** f\_sin\_fi(**double** **long** del,**double** **long** sin\_om,**long** **double** cos\_H){
64. **return** (cos(del) \* sin\_om)/cos\_H;
65. }
66. **inline** **long** **double** f\_fi(**double** **long** del,**double** **long** om,**long** **double** cos\_sis,**long** **double** phi){
67. **long** **double** cos\_fi = (sin(del) \* cos(phi) - cos(om) \* cos(del) \* sin(phi)) / (sin(acos(cos\_sis)));
68. **long** **double** fi = acos(cos\_fi)\*180/M\_PI;
69. **if**(om > 0){
70. fi = 180 - fi;
71. }**else** **if**(om <0){
72. fi -= 180;
73. }**else**{
74. fi = 0;
75. }
76. //    cout << fi << ',';
77. fi = fi\*M\_PI/180;
78. **return** cos\_fi;
79. }
80. **inline** **long** **double** f\_sin\_H(**long** **double** cos\_phi,**long** **double** cos\_del,**long** **double** cos\_om,**long** **double** del,**long** **double** phi){
81. **return** (cos\_phi \* cos\_del \* cos\_om) + (sin(del) \* sin(phi));
82. }
83. **inline** **long** **double** f\_sin\_H0(**long** **double** del,**long** **double** phi){
84. **return** cos(del - phi);
85. }
86. **inline** **long** **double** f\_del(**long** **double** X,**long** **double** N){
87. //return -23.45 \* cos(360.0/365.0 \* (N + 10));
88. **return** 0.006918 - 0.399912 \* cos(X) + 0.070257 \* sin(X) - 0.006758 \* cos(2 \* X) + 0.000908 \* sin(2 \* X);
89. }
90. **inline** **long** **double** f\_X(**long** **double** N){
91. **return** (2 \* M\_PI \* (N - 1))/365;
92. }
93. **inline** **long** **double** f\_om(**long** **double** lam,**long** **double** E){
94. **return** lam + 15 \* E;
95. }
96. **inline** **long** **double** f\_E(**long** **double** X){
97. **return** (0.0172 + 0.4281 \* cos(X) - 7.3515 \* sin(X) - 3.3495 \* cos(2 \* X) - 9.3619 \* sin(2 \* X))/60;
98. }
99. **inline** **long** **double** f\_I(**long** **double** sin\_H,**long** **double** sin\_H0){
100. **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));
101. }
102. **inline** **long** **double** f\_IDN(**long** **double** I,**long** **double** sin\_H){
103. **return** ((1.323 \* I) / (sin\_H)) - 0.5466;
104. }
105. **inline** **long** **double** f\_ISH(**long** **double** I,**long** **double** IDN,**long** **double** sin\_H){
106. **return** I - (IDN \* sin\_H);
107. }
108. **inline** **long** **double** f\_IBR(**long** **double** IsBR,**long** **double** IbBR,**long** **double** IrBR){
109. **return** IbBR + IsBR + IrBR;
110. }
111. **inline** **long** **double** f\_IbBR(**long** **double** IDN,**long** **double** cos\_si){
112. **return** IDN \* cos\_si;
113. }
114. **inline** **long** **double** f\_IsBR(**long** **double** I,**long** **double** ISH,**long** **double** IoH,**long** **double** cos\_si,**long** **double** sin\_H,**long** **double** B){
115. **return** ISH \* (((I-ISH) / (IoH)) \* ((cos\_si) / (sin\_H)) + (1 + ((I-ISH) / (IoH))) \* ((1 + cos(B)) / (2)));
116. }
117. **inline** **long** **double** f\_IoH(**long** **double** sin\_H,**long** **double** N){
118. **return** ISC \* (1 + 0.033 \* cos((N - 2) \* ((2 \* M\_PI) / (365)))) \* sin\_H;
119. }
120. **inline** **long** **double** f\_cos\_si(**long** **double** del,**long** **double** om,**long** **double** B,**long** **double** R,**long** **double** phi){
121. **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);
122. }
123. **inline** **long** **double** f\_IrBR(**long** **double** I,**long** **double** ro,**long** **double** B){
124. **return** (ro \* I) \* ((1 - cos(B)) / (2));
125. }
126. **inline** **long** **double** f\_ro(**long** **double** NS){
127. **return** 0.55;//new concrete Typical albedo (https://en.wikipedia.org/wiki/Albedo)
128. //    return 0.2 \* (1 - NS) + 0.7 \* NS;
129. }
130. **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){
131. om = ((om) \* M\_PI)/180;
132. **long** **double** X = f\_X(N);
133. //    long double E = f\_E(X);
134. **long** **double** del = f\_del(X,N)/\*,om = f\_om(lam,E)\*/;
135. **long** **double** sin\_H = f\_sin\_H(cos(phi),cos(del),cos(om),del,phi),sin\_H0 = f\_sin\_H0(del,phi);
136. **long** **double** I = f\_I(sin\_H,sin\_H0);
137. **long** **double** IDN = f\_IDN(I,sin\_H),cos\_si = f\_cos\_si(del,om,B,R,phi),IoH = f\_IoH(sin\_H,N);
138. **long** **double** ISH = f\_ISH(I,IDN,sin\_H),ro = f\_ro(0);
139. **long** **double** IbBR = f\_IbBR(IDN,cos\_si),IsBR = f\_IsBR(I,ISH,IoH,cos\_si,sin\_H,B),IrBR = f\_IrBR(I,ro,B);
140. **long** **double** fi = f\_fi(del,om,sin\_H,phi);
141. //    cout << N << ' ' << om/15+12 << ' ' << asin (sin\_fi) \* 180.0 / M\_PI << ' ' << asin (sin\_H) \* 180.0 / M\_PI << endl;
142. //    cout << sin\_fi << endl;
143. \*s\_sin\_H = sin\_H;
144. \*s\_fi = fi;
145. \*s\_IsBR = IsBR;
146. \*s\_IrBR = IrBR;
147. all += IbBR;
148. **return** f\_IBR(IsBR,IbBR,IrBR);
149. }
150. **inline** **long** **double** get\_IBR(**long** **double** N,**long** **double** R,**long** **double** B,**long** **double** phi,**long** **double** lam,**long** **double** om){
151. om = ((om) \* M\_PI)/180;
152. **long** **double** X = f\_X(N);
153. //    long double E = f\_E(X);
154. **long** **double** del = f\_del(X,N)/\*,om = f\_om(lam,E)\*/;
155. **long** **double** sin\_H = f\_sin\_H(cos(phi),cos(del),cos(om),del,phi),sin\_H0 = f\_sin\_H0(del,phi);
156. **long** **double** I = f\_I(sin\_H,sin\_H0);
157. **long** **double** IDN = f\_IDN(I,sin\_H),cos\_si = f\_cos\_si(del,om,B,R,phi),IoH = f\_IoH(sin\_H,N);
158. **long** **double** ISH = f\_ISH(I,IDN,sin\_H),ro = f\_ro(0);
159. **long** **double** IbBR = f\_IbBR(IDN,cos\_si),IsBR = f\_IsBR(I,ISH,IoH,cos\_si,sin\_H,B),IrBR = f\_IrBR(I,ro,B);
160. **long** **double** fi = f\_fi(del,om,sin\_H,phi);
161. //    cout << N << ' ' << om/15+12 << ' ' << asin (sin\_fi) \* 180.0 / M\_PI << ' ' << asin (sin\_H) \* 180.0 / M\_PI << endl;
162. //    all += IbBR;
163. **return** f\_IBR(IsBR,IbBR,IrBR);
164. }
165. **inline** **bool** init(**long** **double**\* phi,**long** **double**\* lam){
166. **int** m = 1;
167. **char** address[500];
168. cout << "input the file name:";
169. cin >> address;
170. cout << "input Latitude:";
171. cin >> \*phi;
172. \*phi = ((\*phi) \* M\_PI)/180;
173. cout << "input Longitude:";
174. cin >> \*lam;
175. \*lam = ((\*lam) \* M\_PI)/180;
176. //    cout << "How many solar do you want to build?";
177. //    cin >> m;
178. **for**(**int** i = 0; i < m; i++){
179. //        pair<int,int> a,b;
180. cout << "input the solar upper left corner:";
181. cin >> ra.F >> ra.S;
182. cout << "input the solar lower right corner:";
183. cin >> rb.F >> rb.S;
184. **for**(**int** i = ra.F; i <= rb.F; i++){
185. **for**(**int** q= ra.S; q <= rb.S; q++){
186. is\_solar[i][q] = 1;
187. }
188. }
189. }
190. fstream file;
191. file.open(address,ios::in);
192. **if**(!file){
193. cout << "can't find file!"<<endl;
194. system("pause");
195. **return** 0;
196. }
197. file >> n;
198. **for** (**int** i = 0; i < n; i++){
199. **for**(**int** q = 0; q < n; q++){
200. file >> s[i][q];
201. }
202. }
203. **for**(**int** i = 0; i < n; i++){
204. **for**(**int** q = 0; q < n; q++){
205. cout << s[i][q] << ' ' ;
206. }
207. cout << endl;
208. }
209. **return** 1;
210. }
211. **void** special\_point(){
212. //    for(int i = -75; i < 75;i += 15){
213. ////        113,174,267,357
214. //        get\_IBR(357,0,0,((25) \* M\_PI)/180,((121) \* M\_PI)/180,i);
215. **for**(**int** i = -75; i <= 75;i +=15){
216. //        113,174,267,357
217. get\_IBR(357,((0) \* M\_PI)/180,((45) \* M\_PI)/180,((40) \* M\_PI)/180,((105) \* M\_PI)/180,i);
218. }
219. //    for(long double i = 0; i < 360; i++){
220. //        for(long double q = 0; q < 360; q++){
221. //            cout << "方位:" << i << ' ' << "傾斜:" << q << ' '<< "能量:" << get\_IBR(365.0,i,q,25.0,121.0) << endl;
222. //        }
223. //    }
224. //    for(int i = 0;i <= 20 ;i += 5){
225. //        all = 0;
226. //        for(int q = -75; q <= 75; q+=15){
227. //            all += get\_IBR(113, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q);
228. //            all += get\_IBR(174, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q);
229. //            all += get\_IBR(267, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q);
230. //            all += get\_IBR(357, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q);
231. //            cout << get\_IBR(113, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q) << endl;
232. //            cout << get\_IBR(174, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q) << endl;
233. //            cout << get\_IBR(267, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q) << endl;
234. //            cout << get\_IBR(357, ((i) \* M\_PI)/180, ((21) \* M\_PI)/180, ((25) \* M\_PI)/180, ((121) \* M\_PI)/180,q) << endl;
235. //        }
236. //        cout << all << ',';
237. //        cout << endl;
238. ////        if(i == 40) i-=5;
239. ////        else if(i == 45) i-=5;
240. //    }
241. }
242. **void** get\_hourse(queue<pii>& qu,pii now,**long** **double** sin\_H,**long** **double** fi){
243. **long** **double** X = ((sun \* cos(asin(sin\_H))) \* sin(fi) \* -1),Y = ((sun \* cos(asin(sin\_H))) \* cos(fi));
244. **int** px,py;
245. **if**(X > now.F)px = 1;
246. **else** px = -1;
247. **if**(Y > now.S)py = 1;
248. **else** py = -1;
249. **if**(X - now.F == 0){
250. **for**(**int** i = now.S; i >=0 && i < n; i += py){
251. qu.push(pii(i,now.F));
252. }
253. }**else** **if**(Y - now.S == 0){
254. **for**(**int** i = now.F; i >=0 && i < n; i += px){
255. qu.push(pii(now.S,i));
256. }
257. }**else**{
258. //Y - now.S = m\*(X - now.F)
259. //Y - m\*X - now.S + m\*now.F = 0
260. **long** **double** m = (Y - now.S) / (X - now.F);
261. //        cout << "debug1:" << m << endl;
262. **int** nx = now.F,ny = now.S;
263. **while**(nx >= 0 && nx < n&& ny >=0 && ny < n){
264. //            cout << nx << ' ' << ny << ' ' << abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) << endl;
265. **while**(ny < n && ny >= 0){
266. **if**(abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) > 0.5){
267. ny += py;
268. }**else**{
269. **break**;
270. }
271. }
272. **for**(**int** i = ny; i >= 0 && i < n ; i += py){
273. //                printf("%d %llf %llf %llf %d\n",i,abs(i - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)),abs(i - m\*nx - now.S + m \* now.F),sqrt(1.0 + pow(m,2.0)),(abs(i - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) <= 0.5 && (nx != now.F || i !=now.S)));
274. //                cout << i << ' ' << abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) << endl;
275. **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)){
276. qu.push(pii(i,nx));
277. }**else** **if**(abs(ny - m\*nx - now.S + m \* now.F) / sqrt(1.0 + pow(m,2.0)) > 0.5){
278. **break**;
279. }
280. }
281. nx += px;
282. }
283. }
284. **return**;
285. }
286. pii is\_out(**long** **double** sin\_H,**long** **double** fi){
287. **int** re = 0,cont = 0;
288. **for**(**int** i = ra.F; i <= rb.F; i++){
289. **for**(**int** q = ra.S; q <= rb.S; q++){
290. **if**(is\_solar[i][q]){
291. cont++;
292. queue<pii> qu;
293. get\_hourse(qu,pii(q,i),sin\_H,fi);
294. **long** **double** lb = asin (sin\_H) \* 180.0 / M\_PI;
295. //                cout << "debug2:" << qu.size() << endl;
296. debug("debug2:",qu.size());
297. **while**(!qu.empty()){
298. pii now = qu.front();
299. //                    cout << "debug3:" << now.F << ' ' << now.S << endl;
300. **long** **double** lx = sqrt( (now.F - i) \* (now.F - i) + (now.S - q) \* (now.S - q) );
301. //                    cout << lb << ' ' << atan((s[now.F][now.S] - s[i][q]) / lx) \* 180.0 / M\_PI << ' ' << (s[now.F][now.S] - s[i][q]) << ' ' << lx << endl;
302. debug(lb,atan((s[now.F][now.S] - s[i][q]) / lx) \* 180.0 / M\_PI,(s[now.F][now.S] - s[i][q]),lx);
303. **if**((s[now.F][now.S] < s[i][q]) || ( lb > atan((s[now.F][now.S] - s[i][q]) / lx) \* 180.0 / M\_PI)){
304. qu.pop();
305. }**else**{
306. **break**;
307. }
308. }
309. **if**(qu.size() == 0){
310. re++;
311. }
312. }
313. }
314. }
315. **return** pii(re,cont);
316. }
317. **void** solve(**long** **double** phi, **long** **double** lam,**long** **double** ans[]){
318. **int** N[5] = {113,174,267,357};
319. **for**(**int** R = 0; R <= 0; R++){
320. **for**(**int** B = 0; B <= 90; B++){
321. **for**(**int** q = 0; q <= 3; q++){
322. **for**(**int** i = -75; i <= 75; i += 15){
323. **long** **double** sin\_H,fi,IsBR,IrBR;
324. //                    cout << "debug3:" << N[q] << ' ' << R << ' ' << B << ' ' << i << endl;
325. **long** **double** IBR = get\_IBR(N[q],((R) \* M\_PI)/180,((B) \* M\_PI)/180,phi,lam,i,&sin\_H,&fi,&IsBR,&IrBR);
326. pii tmp = is\_out(sin\_H,fi);
327. //                    cout << "debug4:" << IBR << ' ' << tmp.F << ' ' << tmp.S << endl;
328. ans[B] += IBR \* tmp.F;
329. //                    ans[B] -= IrBR \* tmp.F;
330. ans[B] += IsBR \* (tmp.S - tmp.F);
331. //                    system("pause");
332. }
333. }
334. //            cout << "debug11:" << ans[B] << endl;
335. //            system("pause");
336. }
337. }
338. **for**(**int** i = 0; i <= 90; i++){
339. cout << ans[i] << ',';
340. }
341. }
342. **void** output(**long** **double** ans[95]){
343. fstream file;
344. file.open("out.csv",ios::out);
345. **for**(**int** i = 0; i <= 90; i++){
346. file << ans[i];
347. file << ',';
348. }
349. file.close();
350. }
351. **int** main(){
352. IOS
353. //    special\_point();
354. **long** **double** phi,lam = -5;
355. **long** **double** ans[95] = {};
356. **if**(!init(&phi,&lam))**return** 0;
357. solve(phi,lam,ans);
358. output(ans);
359. system("pause");
360. **return** 0;
361. }