

lab08

```
$ gcc lab08.c
```

```
$ ./a.out < sa.dat
```

Solution 1:

3	6	1		2	9	4		7	8	5	_
9	2	8		3	7	5		1	4	6	_
5	7	4		1	6	8		2	3	9	_
----- ----- -----											
7	3	2		8	1	6		9	5	4	_
4	8	9		7	5	3		6	2	1	_
1	5	6		4	2	9		8	7	3	_
----- ----- -----											
6	9	3		5	8	2		4	1	7	_
8	1	5		6	4	7		3	9	2	_
2	4	7		9	3	1		5	6	8	_

...

Solution 22:

3	6	1		7	9	4		2	8	5	_
9	2	8		3	5	1		7	4	6	_
5	7	4		6	8	2		1	3	9	_
----- ----- -----											
7	3	2		8	1	6		9	5	4	_
4	8	9		5	7	3		6	2	1	_
1	5	6		4	2	9		8	7	3	_
----- ----- -----											
6	9	7		2	3	5		4	1	8	_
8	4	5		1	6	7		3	9	2	_
2	1	3		9	4	8		5	6	7	_

Total number of solution found: 22.

Total number of solutions found: 22.

utime: 0.049629

score: 82.0

- o. [Output] Program output is incorrect.
- o. [Coding] lab08.c spelling errors: Sloutions(1), columm(1), demended(1), determin(1), downmost(1), requirment(1)
- o. [Format] Program format can be improved.

lab08.c

```
1 // EE231002 Lab08. Finding Sudoku Solutions
2 // 111060023, 黃柏霖
3 // 2022/11/14
4
5 #include <stdio.h>
6
7 #define N 9
8
9 int count = 0;                                // count found solution
10
11 // to solve sudoku
12 void solve_sudoku(int A[N][N], int row, int col);
13 // to determin what can be in A[i][j]
14 int check(int A[N][N], int i, int j, int num);
15 // print sudoku
16 void print_sudoku(int A[N][N]);
17
18 int main(void)
19 {
20     int i, j;                                  // loop control
21     char tmp;                                 // store char temporary
22     int M[N][N] = {0};                         // the sudoku
23
24     for (i = 0; i < N; i++) {                  // read sodoku
25         for (j = 0; j < N; j++) {
26             scanf("%c ", &tmp);                 // store tmp
27             // store char to int, and transfer '.' to 0
28             M[i][j] = (tmp - '0' == '.' - '0') ? 0 : tmp - '0';
29         }
30     }
31     solve_sudoku(M, 0, 0);                     // solve sudoku
32     printf("Total number of solution found: %d.\n"
33           , count);                           // print what's demanded
34     X,X count);                            // print what's demand
ed
',,' should not lead a line
34     return 0;
35 }
36
37 // to print sudoku
```

```

38 // int A[] []: the sudoku
39 // return nothing
40 void print_sudoku(int A[N] [N])
41 {
42     int i, j;
43
44     // print sudoku as the requirment
45     for (i = 0; i < N; i++) {
46         printf("  ");
47         if (i == 3 || i == 6) printf("-----|-----|-----\n  ");
48         for (j = 0; j < N; j++) {
49             if (j == 3 || j == 6) printf(" | ");
50             printf("%d ", A[i] [j]);
51         }
52         printf("\n");
53     }
54 }
55
56 // to solve sudoku, count solution, and print it
57 // int A[N] [N]: the sudoku
58 //      row, col: the row and column of sudoku
59 // return nothing
60 void solve_sudoku(int A[N] [N], int row, int col)
61 {
62     int i;
63
64     // find from up-left to down-right
65     // if not 0, don't fill in number
66     if (A[row] [col] != 0) {
67         // found the right if it's still in sudoku
68         if (col < 8) solve_sudoku(A, row, col + 1);
69         // found the next row if it's still in sudoku
70         else if (row < 8) solve_sudoku(A, row + 1, 0);
71         // every position is searched
72         else {
73             count++;                                // new solution found
74             printf("Solution %d:\n", count);
75             print_sudoku(A);                      // print sudoku
76         }
77
78     }
79
80     This line has more than 80 characters
81
82     if (row == 8 && col == 8) {
83         printf("Solved!\n");
84     }
85
86     if (row < 8) {
87         if (col < 8) {
88             solve_sudoku(A, row, col + 1);
89         }
90         else {
91             solve_sudoku(A, row + 1, 0);
92         }
93     }
94
95     if (row == 8 && col == 8) {
96         printf("Solved!\n");
97     }
98
99     if (row < 8) {
100        if (col < 8) {
101            solve_sudoku(A, row, col + 1);
102        }
103        else {
104            solve_sudoku(A, row + 1, 0);
105        }
106    }
107
108    if (row == 8 && col == 8) {
109        printf("Solved!\n");
110    }
111
112    if (row < 8) {
113        if (col < 8) {
114            solve_sudoku(A, row, col + 1);
115        }
116        else {
117            solve_sudoku(A, row + 1, 0);
118        }
119    }
120
121    if (row == 8 && col == 8) {
122        printf("Solved!\n");
123    }
124
125    if (row < 8) {
126        if (col < 8) {
127            solve_sudoku(A, row, col + 1);
128        }
129        else {
130            solve_sudoku(A, row + 1, 0);
131        }
132    }
133
134    if (row == 8 && col == 8) {
135        printf("Solved!\n");
136    }
137
138    if (row < 8) {
139        if (col < 8) {
140            solve_sudoku(A, row, col + 1);
141        }
142        else {
143            solve_sudoku(A, row + 1, 0);
144        }
145    }
146
147    if (row == 8 && col == 8) {
148        printf("Solved!\n");
149    }
150
151    if (row < 8) {
152        if (col < 8) {
153            solve_sudoku(A, row, col + 1);
154        }
155        else {
156            solve_sudoku(A, row + 1, 0);
157        }
158    }
159
160    if (row == 8 && col == 8) {
161        printf("Solved!\n");
162    }
163
164    if (row < 8) {
165        if (col < 8) {
166            solve_sudoku(A, row, col + 1);
167        }
168        else {
169            solve_sudoku(A, row + 1, 0);
170        }
171    }
172
173    if (row == 8 && col == 8) {
174        printf("Solved!\n");
175    }
176
177    if (row < 8) {
178        if (col < 8) {
179            solve_sudoku(A, row, col + 1);
180        }
181        else {
182            solve_sudoku(A, row + 1, 0);
183        }
184    }
185
186    if (row == 8 && col == 8) {
187        printf("Solved!\n");
188    }
189
190    if (row < 8) {
191        if (col < 8) {
192            solve_sudoku(A, row, col + 1);
193        }
194        else {
195            solve_sudoku(A, row + 1, 0);
196        }
197    }
198
199    if (row == 8 && col == 8) {
200        printf("Solved!\n");
201    }
202
203    if (row < 8) {
204        if (col < 8) {
205            solve_sudoku(A, row, col + 1);
206        }
207        else {
208            solve_sudoku(A, row + 1, 0);
209        }
210    }
211
212    if (row == 8 && col == 8) {
213        printf("Solved!\n");
214    }
215
216    if (row < 8) {
217        if (col < 8) {
218            solve_sudoku(A, row, col + 1);
219        }
220        else {
221            solve_sudoku(A, row + 1, 0);
222        }
223    }
224
225    if (row == 8 && col == 8) {
226        printf("Solved!\n");
227    }
228
229    if (row < 8) {
230        if (col < 8) {
231            solve_sudoku(A, row, col + 1);
232        }
233        else {
234            solve_sudoku(A, row + 1, 0);
235        }
236    }
237
238    if (row == 8 && col == 8) {
239        printf("Solved!\n");
240    }
241
242    if (row < 8) {
243        if (col < 8) {
244            solve_sudoku(A, row, col + 1);
245        }
246        else {
247            solve_sudoku(A, row + 1, 0);
248        }
249    }
250
251    if (row == 8 && col == 8) {
252        printf("Solved!\n");
253    }
254
255    if (row < 8) {
256        if (col < 8) {
257            solve_sudoku(A, row, col + 1);
258        }
259        else {
260            solve_sudoku(A, row + 1, 0);
261        }
262    }
263
264    if (row == 8 && col == 8) {
265        printf("Solved!\n");
266    }
267
268    if (row < 8) {
269        if (col < 8) {
270            solve_sudoku(A, row, col + 1);
271        }
272        else {
273            solve_sudoku(A, row + 1, 0);
274        }
275    }
276
277    if (row == 8 && col == 8) {
278        printf("Solved!\n");
279    }
280
281    if (row < 8) {
282        if (col < 8) {
283            solve_sudoku(A, row, col + 1);
284        }
285        else {
286            solve_sudoku(A, row + 1, 0);
287        }
288    }
289
290    if (row == 8 && col == 8) {
291        printf("Solved!\n");
292    }
293
294    if (row < 8) {
295        if (col < 8) {
296            solve_sudoku(A, row, col + 1);
297        }
298        else {
299            solve_sudoku(A, row + 1, 0);
300        }
301    }
302
303    if (row == 8 && col == 8) {
304        printf("Solved!\n");
305    }
306
307    if (row < 8) {
308        if (col < 8) {
309            solve_sudoku(A, row, col + 1);
310        }
311        else {
312            solve_sudoku(A, row + 1, 0);
313        }
314    }
315
316    if (row == 8 && col == 8) {
317        printf("Solved!\n");
318    }
319
320    if (row < 8) {
321        if (col < 8) {
322            solve_sudoku(A, row, col + 1);
323        }
324        else {
325            solve_sudoku(A, row + 1, 0);
326        }
327    }
328
329    if (row == 8 && col == 8) {
330        printf("Solved!\n");
331    }
332
333    if (row < 8) {
334        if (col < 8) {
335            solve_sudoku(A, row, col + 1);
336        }
337        else {
338            solve_sudoku(A, row + 1, 0);
339        }
340    }
341
342    if (row == 8 && col == 8) {
343        printf("Solved!\n");
344    }
345
346    if (row < 8) {
347        if (col < 8) {
348            solve_sudoku(A, row, col + 1);
349        }
350        else {
351            solve_sudoku(A, row + 1, 0);
352        }
353    }
354
355    if (row == 8 && col == 8) {
356        printf("Solved!\n");
357    }
358
359    if (row < 8) {
360        if (col < 8) {
361            solve_sudoku(A, row, col + 1);
362        }
363        else {
364            solve_sudoku(A, row + 1, 0);
365        }
366    }
367
368    if (row == 8 && col == 8) {
369        printf("Solved!\n");
370    }
371
372    if (row < 8) {
373        if (col < 8) {
374            solve_sudoku(A, row, col + 1);
375        }
376        else {
377            solve_sudoku(A, row + 1, 0);
378        }
379    }
380
381    if (row == 8 && col == 8) {
382        printf("Solved!\n");
383    }
384
385    if (row < 8) {
386        if (col < 8) {
387            solve_sudoku(A, row, col + 1);
388        }
389        else {
390            solve_sudoku(A, row + 1, 0);
391        }
392    }
393
394    if (row == 8 && col == 8) {
395        printf("Solved!\n");
396    }
397
398    if (row < 8) {
399        if (col < 8) {
400            solve_sudoku(A, row, col + 1);
401        }
402        else {
403            solve_sudoku(A, row + 1, 0);
404        }
405    }
406
407    if (row == 8 && col == 8) {
408        printf("Solved!\n");
409    }
410
411    if (row < 8) {
412        if (col < 8) {
413            solve_sudoku(A, row, col + 1);
414        }
415        else {
416            solve_sudoku(A, row + 1, 0);
417        }
418    }
419
420    if (row == 8 && col == 8) {
421        printf("Solved!\n");
422    }
423
424    if (row < 8) {
425        if (col < 8) {
426            solve_sudoku(A, row, col + 1);
427        }
428        else {
429            solve_sudoku(A, row + 1, 0);
430        }
431    }
432
433    if (row == 8 && col == 8) {
434        printf("Solved!\n");
435    }
436
437    if (row < 8) {
438        if (col < 8) {
439            solve_sudoku(A, row, col + 1);
440        }
441        else {
442            solve_sudoku(A, row + 1, 0);
443        }
444    }
445
446    if (row == 8 && col == 8) {
447        printf("Solved!\n");
448    }
449
450    if (row < 8) {
451        if (col < 8) {
452            solve_sudoku(A, row, col + 1);
453        }
454        else {
455            solve_sudoku(A, row + 1, 0);
456        }
457    }
458
459    if (row == 8 && col == 8) {
460        printf("Solved!\n");
461    }
462
463    if (row < 8) {
464        if (col < 8) {
465            solve_sudoku(A, row, col + 1);
466        }
467        else {
468            solve_sudoku(A, row + 1, 0);
469        }
470    }
471
472    if (row == 8 && col == 8) {
473        printf("Solved!\n");
474    }
475
476    if (row < 8) {
477        if (col < 8) {
478            solve_sudoku(A, row, col + 1);
479        }
480        else {
481            solve_sudoku(A, row + 1, 0);
482        }
483    }
484
485    if (row == 8 && col == 8) {
486        printf("Solved!\n");
487    }
488
489    if (row < 8) {
490        if (col < 8) {
491            solve_sudoku(A, row, col + 1);
492        }
493        else {
494            solve_sudoku(A, row + 1, 0);
495        }
496    }
497
498    if (row == 8 && col == 8) {
499        printf("Solved!\n");
500    }
501
502    if (row < 8) {
503        if (col < 8) {
504            solve_sudoku(A, row, col + 1);
505        }
506        else {
507            solve_sudoku(A, row + 1, 0);
508        }
509    }
510
511    if (row == 8 && col == 8) {
512        printf("Solved!\n");
513    }
514
515    if (row < 8) {
516        if (col < 8) {
517            solve_sudoku(A, row, col + 1);
518        }
519        else {
520            solve_sudoku(A, row + 1, 0);
521        }
522    }
523
524    if (row == 8 && col == 8) {
525        printf("Solved!\n");
526    }
527
528    if (row < 8) {
529        if (col < 8) {
530            solve_sudoku(A, row, col + 1);
531        }
532        else {
533            solve_sudoku(A, row + 1, 0);
534        }
535    }
536
537    if (row == 8 && col == 8) {
538        printf("Solved!\n");
539    }
540
541    if (row < 8) {
542        if (col < 8) {
543            solve_sudoku(A, row, col + 1);
544        }
545        else {
546            solve_sudoku(A, row + 1, 0);
547        }
548    }
549
550    if (row == 8 && col == 8) {
551        printf("Solved!\n");
552    }
553
554    if (row < 8) {
555        if (col < 8) {
556            solve_sudoku(A, row, col + 1);
557        }
558        else {
559            solve_sudoku(A, row + 1, 0);
560        }
561    }
562
563    if (row == 8 && col == 8) {
564        printf("Solved!\n");
565    }
566
567    if (row < 8) {
568        if (col < 8) {
569            solve_sudoku(A, row, col + 1);
570        }
571        else {
572            solve_sudoku(A, row + 1, 0);
573        }
574    }
575
576    if (row == 8 && col == 8) {
577        printf("Solved!\n");
578    }
579
580    if (row < 8) {
581        if (col < 8) {
582            solve_sudoku(A, row, col + 1);
583        }
584        else {
585            solve_sudoku(A, row + 1, 0);
586        }
587    }
588
589    if (row == 8 && col == 8) {
590        printf("Solved!\n");
591    }
592
593    if (row < 8) {
594        if (col < 8) {
595            solve_sudoku(A, row, col + 1);
596        }
597        else {
598            solve_sudoku(A, row + 1, 0);
599        }
600    }
601
602    if (row == 8 && col == 8) {
603        printf("Solved!\n");
604    }
605
606    if (row < 8) {
607        if (col < 8) {
608            solve_sudoku(A, row, col + 1);
609        }
610        else {
611            solve_sudoku(A, row + 1, 0);
612        }
613    }
614
615    if (row == 8 && col == 8) {
616        printf("Solved!\n");
617    }
618
619    if (row < 8) {
620        if (col < 8) {
621            solve_sudoku(A, row, col + 1);
622        }
623        else {
624            solve_sudoku(A, row + 1, 0);
625        }
626    }
627
628    if (row == 8 && col == 8) {
629        printf("Solved!\n");
630    }
631
632    if (row < 8) {
633        if (col < 8) {
634            solve_sudoku(A, row, col + 1);
635        }
636        else {
637            solve_sudoku(A, row + 1, 0);
638        }
639    }
640
641    if (row == 8 && col == 8) {
642        printf("Solved!\n");
643    }
644
645    if (row < 8) {
646        if (col < 8) {
647            solve_sudoku(A, row, col + 1);
648        }
649        else {
650            solve_sudoku(A, row + 1, 0);
651        }
652    }
653
654    if (row == 8 && col == 8) {
655        printf("Solved!\n");
656    }
657
658    if (row < 8) {
659        if (col < 8) {
660            solve_sudoku(A, row, col + 1);
661        }
662        else {
663            solve_sudoku(A, row + 1, 0);
664        }
665    }
666
667    if (row == 8 && col == 8) {
668        printf("Solved!\n");
669    }
670
671    if (row < 8) {
672        if (col < 8) {
673            solve_sudoku(A, row, col + 1);
674        }
675        else {
676            solve_sudoku(A, row + 1, 0);
677        }
678    }
679
680    if (row == 8 && col == 8) {
681        printf("Solved!\n");
682    }
683
684    if (row < 8) {
685        if (col < 8) {
686            solve_sudoku(A, row, col + 1);
687        }
688        else {
689            solve_sudoku(A, row + 1, 0);
690        }
691    }
692
693    if (row == 8 && col == 8) {
694        printf("Solved!\n");
695    }
696
697    if (row < 8) {
698        if (col < 8) {
699            solve_sudoku(A, row, col + 1);
700        }
701        else {
702            solve_sudoku(A, row + 1, 0);
703        }
704    }
705
706    if (row == 8 && col == 8) {
707        printf("Solved!\n");
708    }
709
710    if (row < 8) {
711        if (col < 8) {
712            solve_sudoku(A, row, col + 1);
713        }
714        else {
715            solve_sudoku(A, row + 1, 0);
716        }
717    }
718
719    if (row == 8 && col == 8) {
720        printf("Solved!\n");
721    }
722
723    if (row < 8) {
724        if (col < 8) {
725            solve_sudoku(A, row, col + 1);
726        }
727        else {
728            solve_sudoku(A, row + 1, 0);
729        }
730    }
731
732    if (row == 8 && col == 8) {
733        printf("Solved!\n");
734    }
735
736    if (row < 8) {
737        if (col < 8) {
738            solve_sudoku(A, row, col + 1);
739        }
740        else {
741            solve_sudoku(A, row + 1, 0);
742        }
743    }
744
745    if (row == 8 && col == 8) {
746        printf("Solved!\n");
747    }
748
749    if (row < 8) {
750        if (col < 8) {
751            solve_sudoku(A, row, col + 1);
752        }
753        else {
754            solve_sudoku(A, row + 1, 0);
755        }
756    }
757
758    if (row == 8 && col == 8) {
759        printf("Solved!\n");
760    }
761
762    if (row < 8) {
763        if (col < 8) {
764            solve_sudoku(A, row, col + 1);
765        }
766        else {
767            solve_sudoku(A, row + 1, 0);
768        }
769    }
770
771    if (row == 8 && col == 8) {
772        printf("Solved!\n");
773    }
774
775    if (row < 8) {
776        if (col < 8) {
777            solve_sudoku(A, row, col + 1);
778        }
779        else {
780            solve_sudoku(A, row + 1, 0);
781        }
782    }
783
784    if (row == 8 && col == 8) {
785        printf("Solved!\n");
786    }
787
788    if (row < 8) {
789        if (col < 8) {
790            solve_sudoku(A, row, col + 1);
791        }
792        else {
793            solve_sudoku(A, row + 1, 0);
794        }
795    }
796
797    if (row == 8 && col == 8) {
798        printf("Solved!\n");
799    }
800
801    if (row < 8) {
802        if (col < 8) {
803            solve_sudoku(A, row, col + 1);
804        }
805        else {
806            solve_sudoku(A, row + 1, 0);
807        }
808    }
809
810    if (row == 8 && col == 8) {
811        printf("Solved!\n");
812    }
813
814    if (row < 8) {
815        if (col < 8) {
816            solve_sudoku(A, row, col + 1);
817        }
818        else {
819            solve_sudoku(A, row + 1, 0);
820        }
821    }
822
823    if (row == 8 && col == 8) {
824        printf("Solved!\n");
825    }
826
827    if (row < 8) {
828        if (col < 8) {
829            solve_sudoku(A, row, col + 1);
830        }
831        else {
832            solve_sudoku(A, row + 1, 0);
833        }
834    }
835
836    if (row == 8 && col == 8) {
837        printf("Solved!\n");
838    }
839
840    if (row < 8) {
841        if (col < 8) {
842            solve_sudoku(A, row, col + 1);
843        }
844        else {
845            solve_sudoku(A, row + 1, 0);
846        }
847    }
848
849    if (row == 8 && col == 8) {
850        printf("Solved!\n");
851    }
852
853    if (row < 8) {
854        if (col < 8) {
855            solve_sudoku(A, row, col + 1);
856        }
857        else {
858            solve_sudoku(A, row + 1, 0);
859        }
860    }
861
862    if (row == 8 && col == 8) {
863        printf("Solved!\n");
864    }
865
866    if (row < 8) {
867        if (col < 8) {
868            solve_sudoku(A, row, col + 1);
869        }
870        else {
871            solve_sudoku(A, row + 1, 0);
872        }
873    }
874
875    if (row == 8 && col == 8) {
876        printf("Solved!\n");
877    }
878
879    if (row < 8) {
880        if (col < 8) {
881            solve_sudoku(A, row, col + 1);
882        }
883        else {
884            solve_sudoku(A, row + 1, 0);
885        }
886    }
887
888    if (row == 8 && col == 8) {
889        printf("Solved!\n");
890    }
891
892    if (row < 8) {
893        if (col < 8) {
894            solve_sudoku(A, row, col + 1);
895        }
896        else {
897            solve_sudoku(A, row + 1, 0);
898        }
899    }
900
901    if (row == 8 && col == 8) {
902        printf("Solved!\n");
903    }
904
905    if (row < 8) {
906        if (col < 8) {
907            solve_sudoku(A, row, col + 1);
908        }
909        else {
910            solve_sudoku(A, row + 1, 0);
911        }
912    }
913
914    if (row == 8 && col == 8) {
915        printf("Solved!\n");
916    }
917
918    if (row < 8) {
919        if (col < 8) {
920            solve_sudoku(A, row, col + 1);
921        }
922        else {
923            solve_sudoku(A, row + 1, 0);
924        }
925    }
926
927    if (row == 8 && col == 8) {
928        printf("Solved!\n");
929    }
930
931    if (row < 8) {
932        if (col < 8) {
933            solve_sudoku(A, row, col + 1);
934        }
935        else {
936            solve_sudoku(A, row + 1, 0);
937        }
938    }
939
940    if (row == 8 && col == 8) {
941        printf("Solved!\n");
942    }
943
944    if (row < 8) {
945        if (col < 8) {
946            solve_sudoku(A, row, col + 1);
947        }
948        else {
949            solve_sudoku(A, row + 1, 0);
950        }
951    }
952
953    if (row == 8 && col == 8) {
954        printf("Solved!\n");
955    }
956
957    if (row < 8) {
958        if (col < 8) {
959            solve_sudoku(A, row, col + 1);
960        }
961        else {
962            solve_sudoku(A, row + 1, 0);
963        }
964    }
965
966    if (row == 8 && col == 8) {
967        printf("Solved!\n");
968    }
969
970    if (row < 8) {
971        if (col < 8) {
972            solve_sudoku(A, row, col + 1);
973        }
974        else {
975            solve_sudoku(A, row + 1, 0);
976        }
977    }
978
979    if (row == 8 && col == 8) {
980        printf("Solved!\n");
981    }
982
983    if (row < 8) {
984        if (col < 8) {
985            solve_sudoku(A, row, col + 1);
986        }
987        else {
988            solve_sudoku(A, row + 1, 0);
989        }
990    }
991
992    if (row == 8 && col == 8) {
993        printf("Solved!\n");
994    }
995
996    if (row < 8) {
997        if (col < 8) {
998            solve_sudoku(A, row, col + 1);
999        }
1000       else {
1001           solve_sudoku(A, row + 1, 0);
1002       }
1003   }
1004 }
```

```

77 }
78 // if 0, then try which number can fill in
79 else {
80     for (i = 1; i <= N; i++) {
81         if (check(A, row, col, i)) {
82             A[row][col] = i;                                // try through 1 to 9
83             // not searching to the rightmost yet
84             if (col < 8) {
85                 solve_sudoku(A, row, col + 1);           // found the right
86                 A[row][col] = 0;                          // trace back
87             }
88             // not searching to the downmost yet
89             else if (row < 8) {
90                 solve_sudoku(A, row + 1, 0);           // found the next row
91                 A[row][col] = 0;                          // trace back
92             }
93             // every position is searched
94             else {
95                 count++;                             // new solution found
96                 printf("Solution %d:\n", count);
97                 print_sudoku(A);                     // print sudoku
98                 A[row][col] = 0;                      // trace back
99             }
100        }
101    }
102 }
103 }
104
105 // to check whether the num can be put in A[row][col]
106 // int A[N][N]: the sudoku
107 //      row, col: the tow and column of sudoku
108 //      num: the number that is checking
109 // return whether num can be put in
110 int check(int A[N][N], int row, int col, int num)
111 {
112     int k;                                         // loop control
113     int cell_x0 = row / 3 * 3;                   // the leftmost x of a cell
114     int cell_y0 = col / 3 * 3;                   // the uppermost y of a cell
115
116     for (k = 0; k < N; k++) {
117         if (A[row][k] == num) return 0;            // check row

```

```
118     if (A[k][col] == num) return 0;           // check column
119     if (A[cell_x0 + k % 3][cell_y0 + k / 3] == num)
120         return 0;                           // check cell
121     }
122     return 1;                            // num is available
123 }
```