```
#include<stdio.h>
#include<strings.h>

/* This program was complied on Unix (Solaris) using the compiler "gcc". */

#define ONE "100000000\0"
#define TWO "010000000\0"
#define THREE "001000000\0"
#define FOUR "000100000\0"
#define FIVE "000010000\0"
#define SIX "000001000\0"
#define SEVEN "000000100\0"
#define EIGHT "000000010\0"
#define NINE "000000001\0"
#define BIG_X "111111111\0"

char all[10][10][10];
int all_wt[10][10];

/* all[i][j][k] stores the values that are valid choices at every cell of the
9x9 Sudoku array.
The first and second indices to the array all[][][], i and j in above example,
respectively
specify the row and column numbers. Only i and j values from 1 to 9 are used for
all[i][j][].
For a particular row and column, i.e., at a particular value of i and j in the
above example,
we have a 10-element character array. Since a valid string must end with an end-
of-string character,
deonoted by '\0', the tenth character in the array, i.e., k=9 in the above
example, is
initialized to end-of-string. The remaining entries, i.e., for k=0 to k=8, are
respectively
used to denote whether the values 1, 2, ..., 9, are valid choices for that cell
of the
Sudoku array. (Important to note the shift, as k=0 corresponds to value 1 in the
cell, k=1
corresponds to value 2 in the cell, and so on.)

all_wt[i][j] array stores the number of possible values that remain as valid
choices for
the value at the i-th row and j-th column. A weight value of 1 denotes that only
one value
is a valid choice for that cell. A weight value higher than 1 deontes that there
are still multiple
value choices for that cell. What does a weight value of 0 denote?

For example, if we know that cell at row 3 and column 4 is assigned the value 8,
then all[3][4][7]
will be '1'; all[3][4][k] values will be '0' for k=0, 1, 2, 3, 4, 5, 6, 8, and
all[3][4][9] will
be '\0'; furthermore wt_all[3][4] will be 1.

Similarly, if we know that the cell at row 7 and column 3 has a choice of values
1 and 7, then
all[7][3][0] and all[7][3][6] will be '1'; all[7][3][k] values will be '0' for
k= 1, 2, 3, 4, 5,
```

```
7, 8, and all[7][3][9] will be '\0'; furthermore wt_all[7][3] will be 2.

IMPORTANT: Whenever you remove a value choice from a particular cell in the
Sudoku array,
make sure that you suitably update the weight value for that cell in the array.
For example, if you
discover that the value 7 is no longer a valid choice for the cell at row 7 and
col 3 in the above
example, then you can carry out the following update to indicate that.
        all[3][7][6]='0'
In such a case, as the previous value of all[7][3][6] was '1', then you MUST
decrement by 1
the value of all_wt[7][3].
*/

FILE *fpin;

main(argc, argv)
int argc;
char **argv;
{


char line[20];
int row, col, val;
int mupdt;
int a, b, c;

/* Filename for input puzzle given as the second argument in the command-line
when the program is
run in Unix. Hence, if you compile your program into a file with name a.out,
then typing
      a.out example-puzzle
will cause your program to take the puzzle in the file named example-puzzle as
input.
An example puzzle must be written as a text file where each line has three
integer entries,
      row col cell-value
Of course, row and col each take values from 1 to 9, giving a maximum of 81
lines in the file.
The cell-value can be 1, 2, 3, 4, 5, 6, 7, 8, or 9 for a cell for which the
value is given in
specified by the original puzzle. On the other hand, if the value at a
particular row and col is
unspecified in the original puzzle, then cell-value of 10 can be used.
*/

if(open_files(argc, argv)==0){
   exit(-1);
}

initialize_all();

while(freadline(fpin, line)!=EOF){
   sscanf(line, "%d%d%d", &row, &col, &val);
   if((val<1)||(val>9)){
      continue;
```

```
   }
   else{
      fprintf(stdout, "row: %d col: %d val: *%d*\n", row, col, val);
      set_all_i_j_to_fully_spec_val(row, col, val);
   }

}
print_all();

}

/* Opens the puzzle file for reading the puzzle. */
int open_files(numstrngs, strngs)
int numstrngs;
char **strngs;
{
if(numstrngs!=2){
        fprintf(stdout, "Usage: %s input-file-name\n", strngs[0]);
        return(0);
}
if((fpin= fopen(strngs[1], "r"))== NULL){
  fprintf(stdout, "ERROR: can't open file %s for reading\n", strngs[1]);
  fprintf(stderr, "ERROR: can't open file %s for reading\n", strngs[1]);
  return(0);
}
return(1);

}

/* Initializes all[][][] and wt_all[][] in the manner described above
(near the declaration of these two arrays).
*/

initialize_all()
{
int i, j;


for(i=1; i<=9; i++){
   for(j=1; j<=9; j++){
      strcpy(all[i][j],BIG_X);
      all_wt[i][j]=9;
   }
}
}

set_all_i_j_to_fully_spec_val(row, col, val)
int row, col, val;
{
int i;

if((val<1)||(val>9)){
   fprintf(stderr, "At this stage, each entry read should be fully-specified,
i.e., between 1 and 9.\n");
   return(-1);
}
for(i=0; i<9; i++){
```

```
   if(i==(val-1)){
      if(all[row][col][i]!='1'){
         return(-1);
      }
   }
   else{
      all[row][col][i]='0';
   }
}
all_wt[row][col]=1;
return(1);
}

/* Use this to print the current status of all the entries in the Sudoku array.
Each cell with a fully specified value is shown as an integer sandwiched between
two
asterix, such as *4*. A cell where multiple values are possible, a nine-
character string
is printed. For example, 1_34_____ indicates that values 1, 3, and 4 are
possible. Each value
that is no longer possible is shown using the underscore character.
*/

print_all()
{
int i, j, k;
char binary_ver[10];
int fully_spec_val;

for(i=1; i<=9; i++){
   for(j=1; j<=9; j++){
      if(all_wt[i][j]==1){
         fully_spec_val=fully_spec_val_in_all(i,j);
         fprintf(stdout, "   *%1d*    ", fully_spec_val);
      }
      else{
         for(k=0; k<9; k++){
            if(all[i][j][k]=='1'){
               fprintf(stdout, "%1d", k+1);
            }
            else{
               fprintf(stdout, "_");
            }
         }
         fprintf(stdout, " ", all[i][j]);
         /*
         fprintf(stdout, "%9s ", all[i][j]);
         */
      }
      if((j==3)||(j==6)){
         fprintf(stdout, "| ");
      }
   }
   fprintf(stdout, "\n");
   if((i==3)||(i==6)){
      for(k=0; k<89; k++){
         fprintf(stdout, "-");
```

```
      }
      fprintf(stdout, "\n");
   }

}
}

/* Use this print how close you are to solving the puzzle. In a completely
solved puzzle, every one
of its 81 cells has a single value assigned.
*/

print_summary()
{
int i, j;
int wt_freq[10];
int wt_ij;
for(i=1; i<=9; i++){
   wt_freq[i]=0;
}
for(i=1; i<=9; i++){
   for(j=1; j<=9; j++){
      wt_ij=all_wt[i][j];
      wt_freq[wt_ij]++;
   }
}
printf("Summary of final weights: ");
for(i=1; i<=9; i++){
   printf("%d ", wt_freq[i]);
}
if(wt_freq[1]==81){
   printf("Solved.\n");
}
else{
   printf("NOT completely solved.\n");
}

}


fully_spec_val_in_all(i,j)
int i, j;
{
int k;
int retval;

retval= (-1);

for(k=0; k<9; k++){
   if(all[i][j][k]=='1'){
      if(retval!= (-1)){
         return(-1);
      }
      else{
         retval=k+1;
      }
   }
```

```
}
return(retval);
}

readline(char *str)
{
int i=0;
char ch;

while(scanf("%c", &ch)!=EOF){
   if(ch!='\n'){
      str[i]=ch;
      i++;
   }
   else{
      str[i]= '\0';
      return(1);
   }
}
return(EOF);
}

freadline(fp, str)
FILE *fp;
char *str;
{
int i=0;
char ch;

while(fscanf(fp, "%c", &ch)!=EOF){
   if(ch!='\n'){
      str[i]=ch;
      i++;
   }
   else{
      str[i]= '\0';
      return(1);
   }
}
return(EOF);
}
```