CSE 251 - more Arrays

Arend Hintze

A H											
A N	3	1	3	2	5	7	0	7	0	1	29
A R	3	3	0	3	5	0	6	3	0	1	24
C R	3	3	2	5	5	0	2	0	0	0	20
D K	2	3	0	0	2	7	3	3	9	1	30
D M	3	3	5	1	4	7	0	6	9	1	39
D S	3	3	5	1	5	7	4	7	9	1	45
D Z	3	3	5	5	5	7	7	7	9	1	52
F L	3	3	0	3	5	2	6	2	2	1	22
G J	3	3	0	2	2	5	6	0	0	1	22
G W	3	3	5	5	5	7	7	7	9	1	52
H K	2	3	1	0	2	0	6	0	0	1	15
J G	3	3	0	0	3	0	0	0	0	1	10
J L	3	3	5	5	2	6	7	5	0	1	37
J M	3	3	5	5	3	7	5	7	9	1	48
J S	3	3	4	5	3	0	5	5	0	1	29
J W	3	3	5	3	5	7	7	7	9	1	50
K W	3	3	5	5	4	7	7	7	9	1	51
M J	3	3	3	0	2	0	7	0	0	1	19
M R	3	3	5	5	5	7	7	0	0	1	36
M S	3	3	0	1	2	0	7	0	0	1	17
N H	3	3	3	5	2	7	7	7	9	1	47
R B	3	3	0	0	4	6	6	6	0	1	29
R L	3	3	0	0	0	0	0	0	0	1	7
R S	3	3	1	0	2	0	0	7	9	1	26
S G											
T P	3	3	1	2	5	7	6	0	0	1	28
T S	3	3	5	5	5	7	7	7	9	1	52
X Z	3	1	0	0	0	0	2	5	0	1	12
Y J	3	3	4	5	5	5	7	7	9	1	49

A	28-30
B	24-27
C	19-23
D	15-18
F	0-14

```
void emptyBoxSizeNxM(int N,int M){
    int i,j;
    void line(void){
        //something
    }
    void border(void){
        //something
    }
    line();
    border();
```

NO!

no function definitions within functions

```
void line(void){
    //something
}
void border(void){
    //something
}
void emptyBoxSizeNxM(int N,int M){
    int i,j;
    line();
    border();
```

YES!

```
//question 8
int getCountOfNrBelowM(int N[100], int M){
    /* you get the array N
   of size 100 filled with numbers,
     return the count of numbers below M
     */
    int i,count=0;
    for(i=1;i<101;i++)
        if(N[i]<M)
            count++;
    return count;
}
```

NO!

i will exceed Ns array boundaries

```
//question 8
int getCountOfNrBelowM(int N[100], int M){
    /* you get the array N
of size 100 filled with numbers,
     return the count of numbers below M
     */
    int i,count=0;
    for(i=0;i<100;i++)
        if(N[i]<M)
            count++;
    return count;
}
```

YES!

compile compile compile

- code that is not compiling is worth nothing!
- next exam: every compilation error is -1 point

its all about style!

```
//question 4
void dashRightSizeN(int N){
     /*make a / from * */
     int i,j;
     for(i=0;i<N;i++){
     for(j=0;j<N-i;j++)
               printf(" ");
          for(j=0;j<N;j++)
                    printf("*");
          printf("\n");}
}
```

```
//question 4
void dashRightSizeN(int N){
     /*make a / from * */
     int i,     j;
     for(i=0;i< N;i++){

     for(j=0;j<N-i;
j++)
```

```
                    printf(" ");
                       for(j=0;j<N;j++)
                 printf("*");
printf("\n");}
}
```

style!

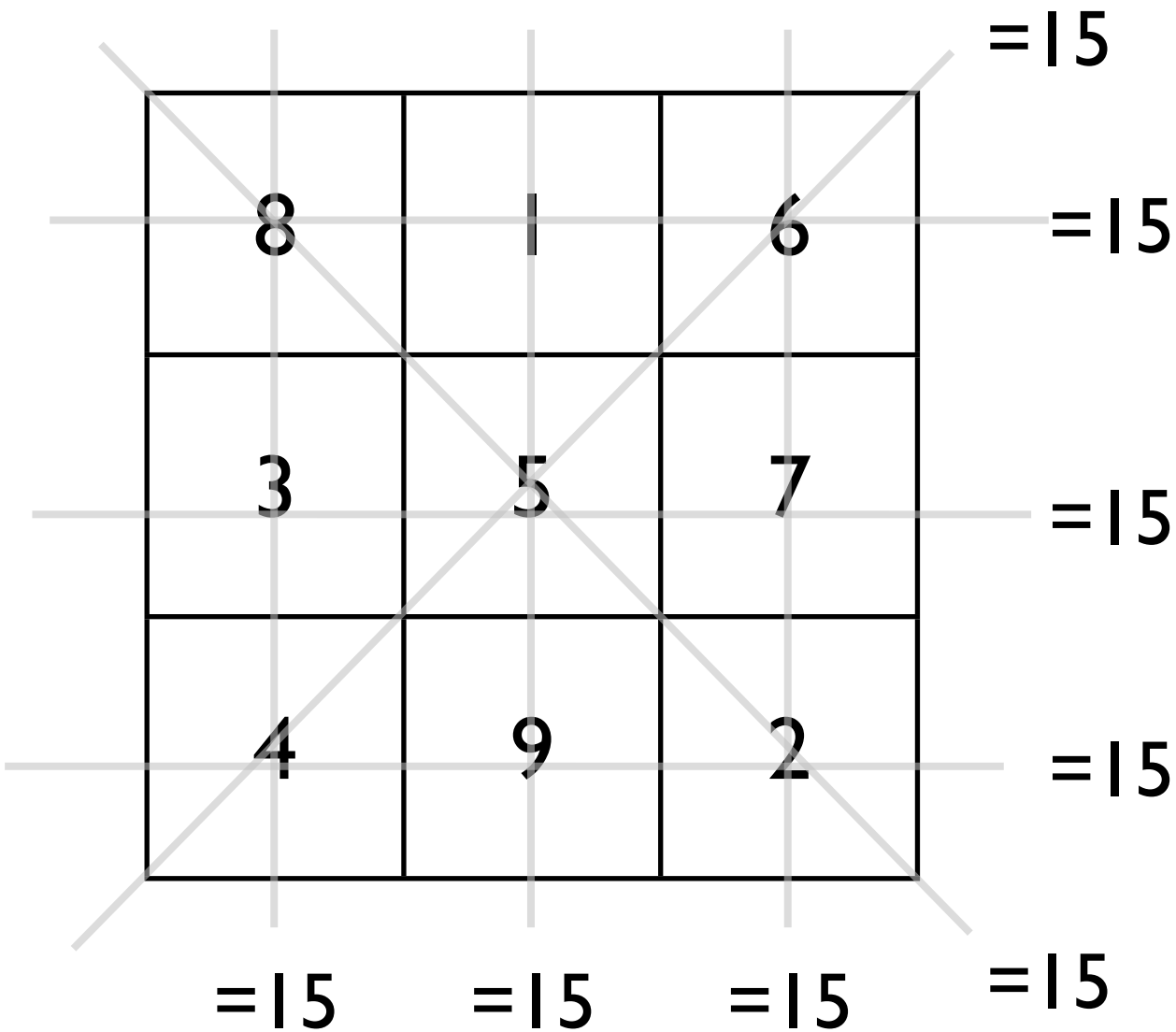
- we indent using a tab after EACH{
- we indent using a tab after we could have used a {
- we insert blank lines to make code more readable, not less readable
- if we insert a blank line, we could also write a comment
- each } deserves a new line

magic squares

8	1	6
3	5	7
4	9	2

I will ask for 4x4, 5x5, and maybe 6x6
(4x4 sum is 34, 5x5 sum is 65, 6x6 is 111)

magic squares



I will ask for 4x4, 5x5, and maybe 6x6
(4x4 sum is 34, 5x5 sum is 65, 6x6 is 111)

things we know

- we will need a 3x3 array
- we should be able to test if a matrix is magic
- keep making random ones until we find one
- we know one that works, maybe we use it for testing

in case that is too simple for you:
http://www.math.hmc.edu/funfacts/ffiles/10001.4-8.shtml#

```
#define Dim 3
#define Sum 15

bool isMagic(int M[Dim][Dim]){
    int i,j,k;
    //check rows
    for(i=0;i<Dim;i++){
        k=0;
        for(j=0;j<Dim;j++)
            k+=M[i][j];
        if(k!=Sum)
            return false;
    }
    //check columns
    for(i=0;i<Dim;i++){
        k=0;
        for(j=0;j<Dim;j++)
            k+=M[j][i];
        if(k!=Sum)
            return false;
    }
    //check diagonal 1
    k=0;
    for(i=0;i<Dim;i++)
        k+=M[i][i];
    if(k!=Sum)
        return false;
    //check diagonal 2
    k=0;
    for(i=0;i<Dim;i++)
        k+=M[i][Dim-1-i];
    if(k!=Sum)
        return false;
    return true;
}
```

how to test this?

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

#define Dim 3
#define Sum 15

bool isMagic(int M[Dim][Dim]);

int main(int argc, const char * argv[]){
    int Solution[3][3]={{8,1,6},{3,5,7},{4,9,2}};
    if(isMagic(Solution))
        printf("magic!\n");
    else
        printf("no magic\n");
...
```

next: making a magic square

```
for(i=0;i<Dim;i++)
    for(j=0;j<Dim;j++)
        M[i][j]=1+(rand()%9);

for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

```
do{
    for(i=0;i<Dim;i++)
        for(j=0;j<Dim;j++)
            M[i][j]=1+(rand()%9);
} while(!isMagic(M));
```

```
for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

yes, instead of:

```
while(){
}
```

you can say:

```
do{
}while();
```

```
do{
    for(i=0;i<Dim;i++)
        for(j=0;j<Dim;j++)
            M[i][j]=1+(rand()%9);
} while(!isMagic(M));
```

yes, instead of:

```
while(){
}
```

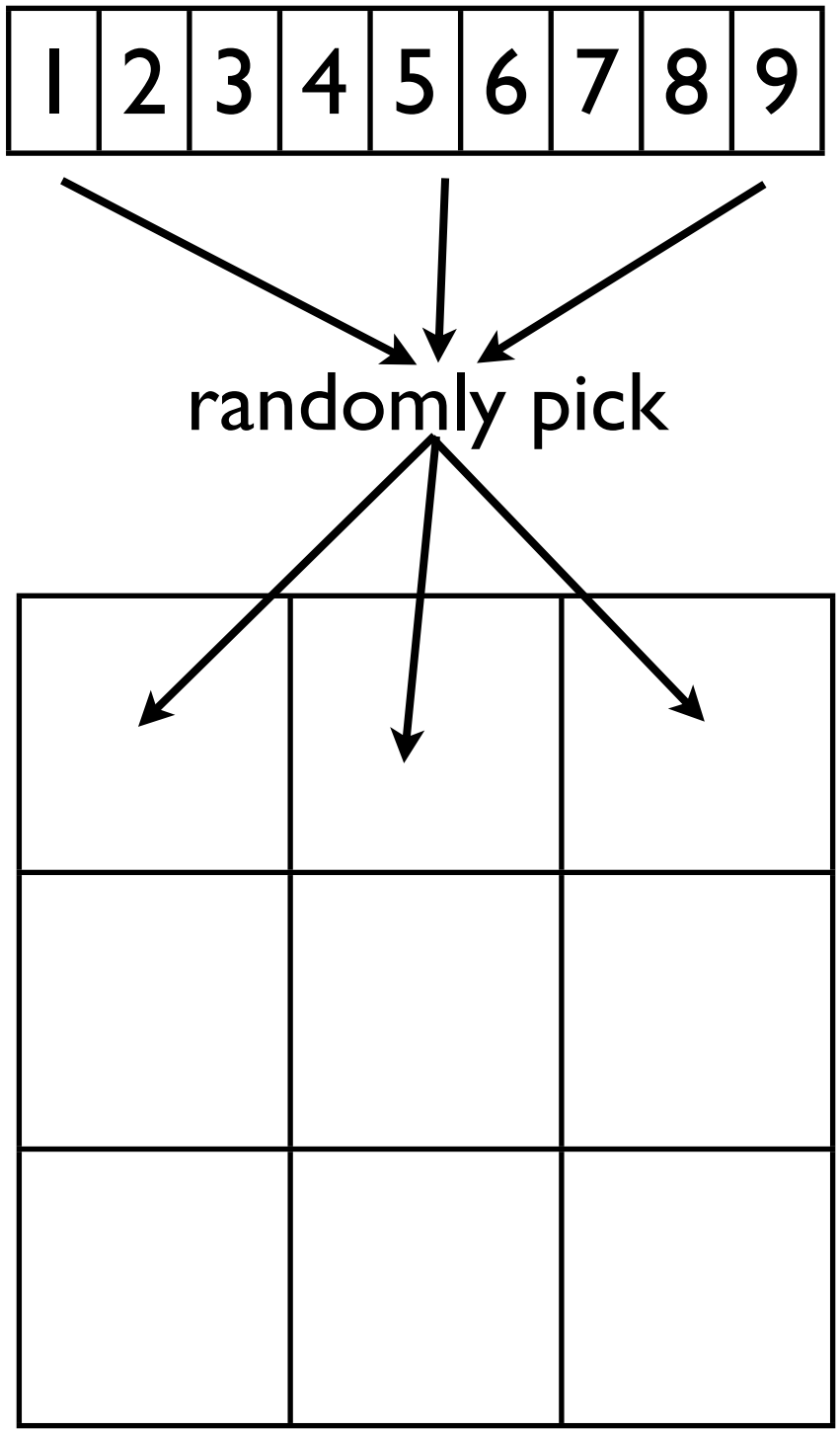
you can say:

```
do{
}while();
```

```
for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

5 4 6
6 5 4
4 6 5

Wait, what is wrong?



1	2	3	4	5	6	7	8	9

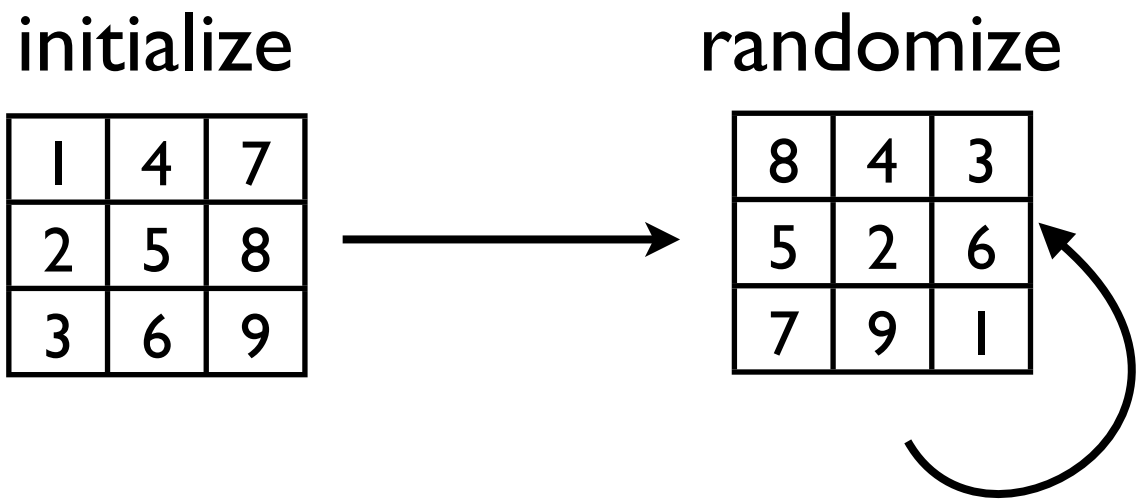
1) randomize list

8	4	3	5	2	6	7	9	1

2) pick in order

8	4	3
5	2	6
7	9	1

or:



```
k=1;
for(i=0;i<Dim;i++)
    for(j=0;j<Dim;j++){
        M[i][j]=k;
        k++;
    }
do{
    for(i=0;i<Dim*Dim;i++){
        x1=rand()%Dim;
        y1=rand()%Dim;
        x2=rand()%Dim;
        y2=rand()%Dim;
        j=M[x1][y1];
        M[x1][y1]=M[x2][y2];
        M[x2][y2]=j;
    }
} while(!isMagic(M));
```

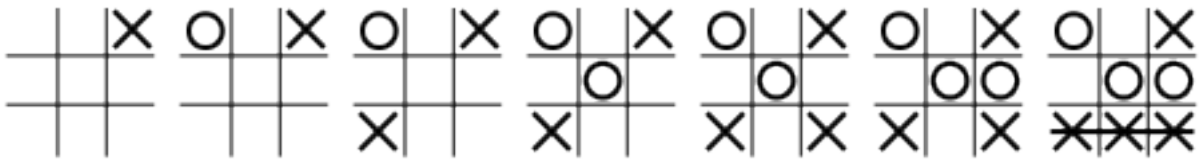
```
6 1 8
7 5 3
2 9 4
```

making a magic square 4x4 might be a question in the next exam...

command line parameters

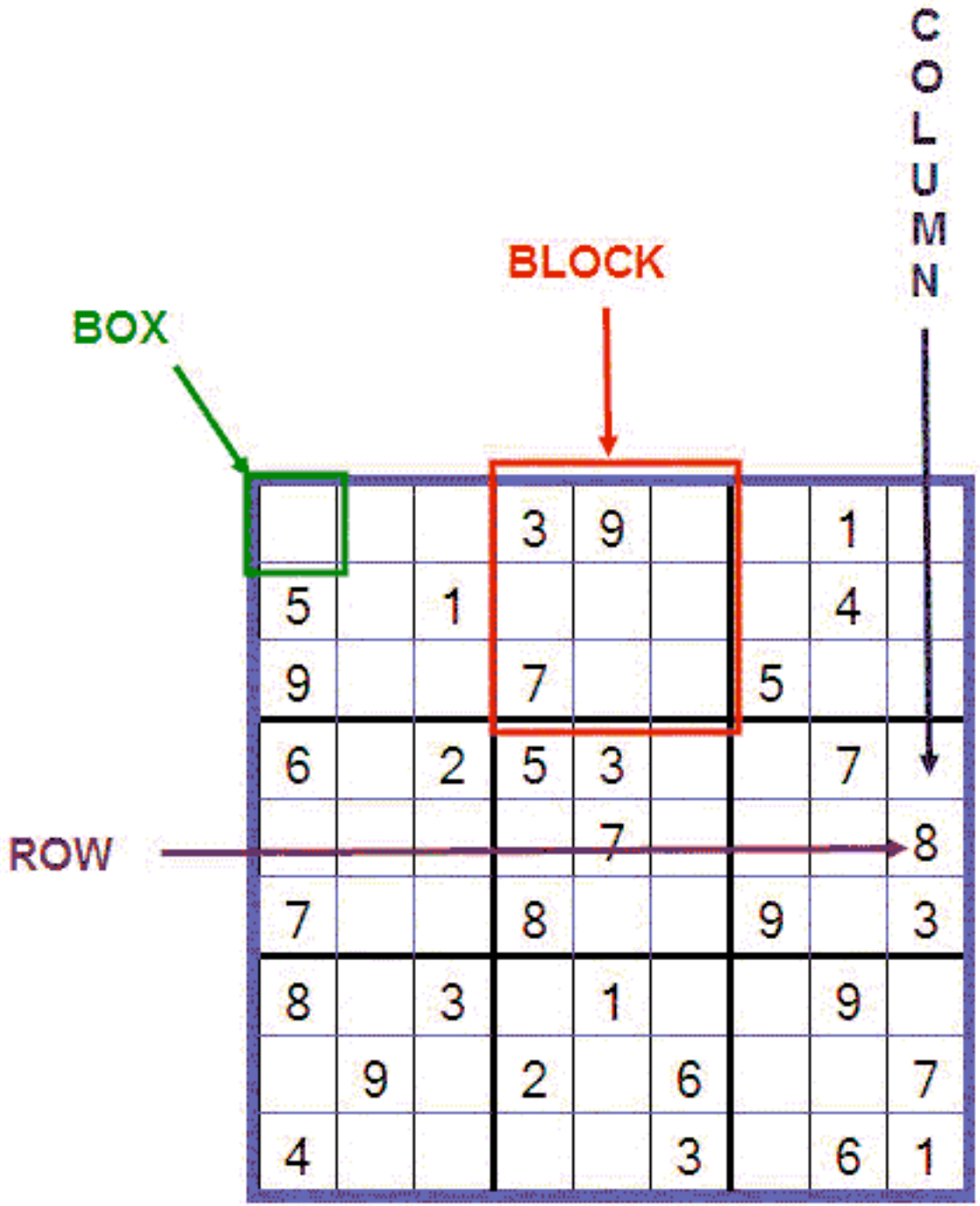
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char *argv[]){
    int i;
    for(i=0;i<argc;i++)
        printf("%i string:%s\n",i,argv[i]);
    return 0;
}

int main(int argc, const char *argv[]){
    int i;
    for(i=0;i<argc;i++)
        printf("%i string:%s int:%i\n",i,argv[i],atoi(argv[i]));
    return 0;
}
```

tic-tac-toe

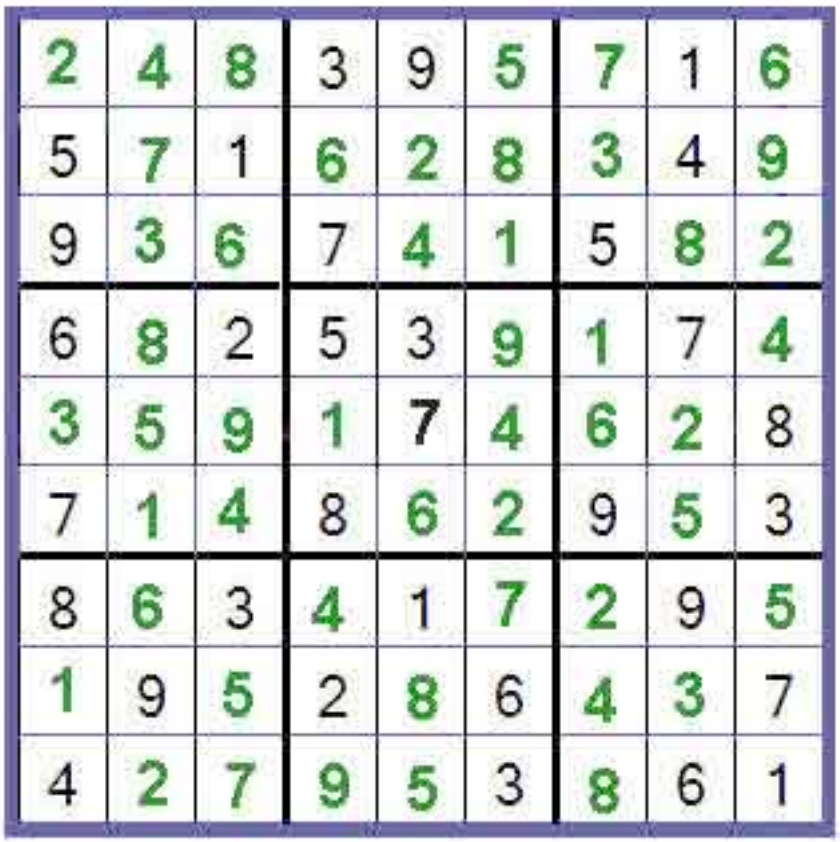


1	0	2
0	1	1
2	2	2

- get a tic tac toe situation from the commandline
- print it
- check if someone won
- print who wins, or no win, or draw
- advanced: make a tic tac toe game against a random opponent



solved if:
all numbers 1-9
in each block
in each column
in each line



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

```c
#include <stdio.h>
#include <stdlib.h>

void showSudoku(int sudoku[9][9]){
    int i,j;
    for(i=0;i<9;i++){
        for(j=0;j<9;j++)
            printf("%i ",sudoku[i][j]);
        printf("\n");
    }
}

int main(int argc, const char *argv[]){
    int i,j,k=1;
    int sudoku[9][9];
    for(i=0;i<9;i++)
        for(j=0;j<9;j++){
            sudoku[i][j]=atoi(argv[k]);
            k++;
        }
    showSudoku(sudoku);
    return 0;
}
```

homework

- you get 81 commandline parameters
- load them in an array
- test the array for being a solved sudoku
- print a SOLVED! if so, otherwise print TRY AGAIN!