Exam Seating Chart Generator

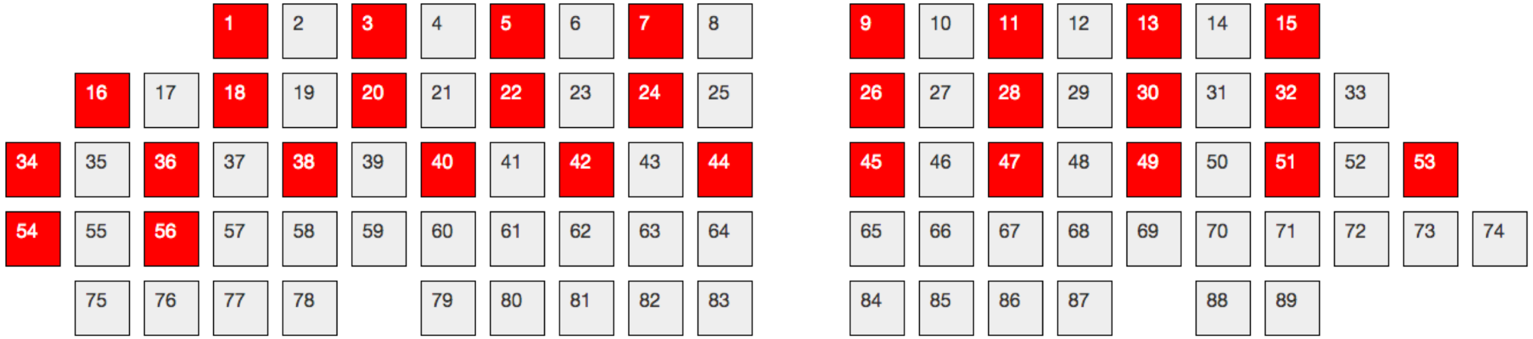
Suhail Ghafoor

Problem:

Generate a seating chart for an exam

Solution:

Enter names or number of students and if there are enough seats then assign everyone a seat.



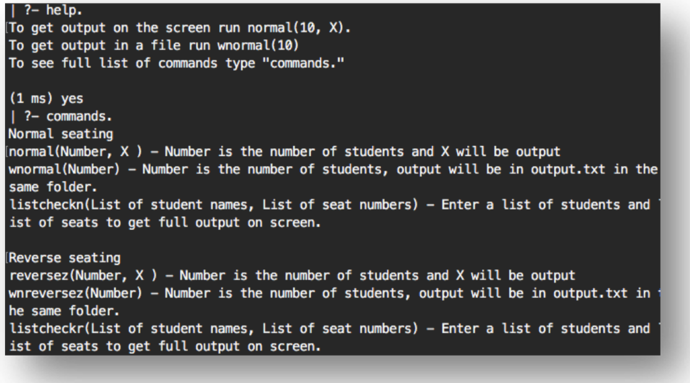
			1	2	**3**	4	**5**	6	**7**	8		**9**	10	**11**	12	**13**	14	**15**			
	16	17	**18**	19	**20**	21	**22**	23	**24**	25		**26**	27	**28**	29	**30**	31	**32**	33		
34	35	**36**	37	**38**	39	**40**	41	**42**	43	**44**		**45**	46	**47**	48	**49**	50	**51**	52	**53**	
54	55	**56**	57	58	59	60	61	62	63	64		65	66	67	68	69	70	71	72	73	74
	75	76	77	78		79	80	81	82	83		84	85	86	87		88	89			

*actual program footage

As a user:

What is the criteria for a good program?

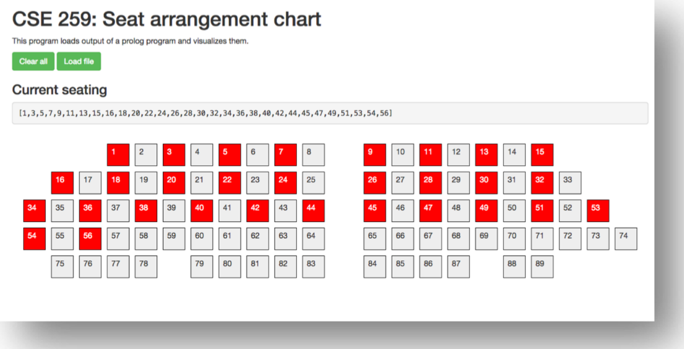
- Easy to set up
  - No external dependencies.
  - No variables to change.
- Easy to use
  - Help or readme file.
  - Options to use it in multiple ways.
- User friendly UI
  - Results are meaningful.
  - Does not require too many steps.



```
| ?- help.
To get output on the screen run normal(10, X).
To get output in a file run wnormal(10)
To see full list of commands type "commands."

(1 ms) yes
| ?- commands.
Normal seating
normal(Number, X ) - Number is the number of students and X will be output
wnormal(Number) - Number is the number of students, output will be in output.txt in the
same folder.
listcheckn(List of student names, List of seat numbers) - Enter a list of students and
ist of seats to get full output on screen.

Reverse seating
reversez(Number, X ) - Number is the number of students and X will be output
wnreversez(Number) - Number is the number of students, output will be in output.txt in
he same folder.
listcheckr(List of student names, List of seat numbers) - Enter a list of students and
ist of seats to get full output on screen.
```



CSE 259: Seat arrangement chart

This program loads output of a prolog program and visualizes them.

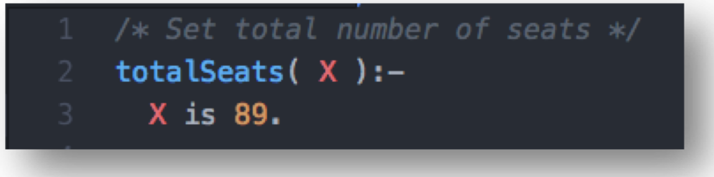
Current seating

```
[1,3,5,7,9,11,13,15,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,45,47,49,51,53,54,56]
```

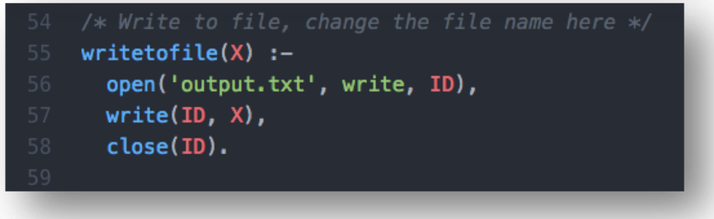
As a programmer:

What is the criteria for good code?

- Easy to modify
  - Well commented
  - Global variables in same place.



```
/* Set total number of seats */
totalSeats( X ):-
  X is 89.
```



```
/* Write to file, change the file name here */
writetofile(X) :-
  open('output.txt', write, ID),
  write(ID, X),
  close(ID).
```

- Easy to extend
  - Modular design.
  - Can connect with other applications.

Problem:

Program designed to take in list of variables needs to take in a single number.

- Helpful when assigning small number of seats but pain when there are a lot of students.

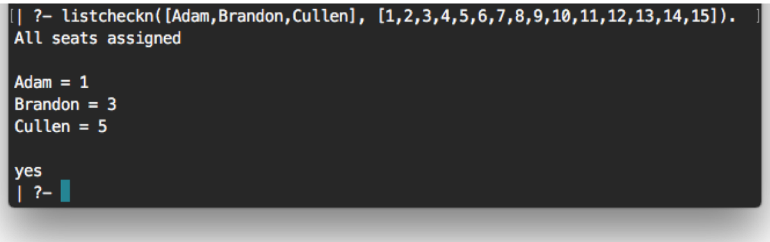
- Default prolog function:

```
length([1,2,3,4,5,6], X).
X = 6

length( X , 4).
X = [Var, Var, Var, Var]
```

#1
BEST COLLEGES
U.S.News
MOST INNOVATIVE 2017
in the U.S. for innovation
#1 ASU #2 Stanford #3 MIT
– U.S. News & World Report, 2016 and 2017

Before

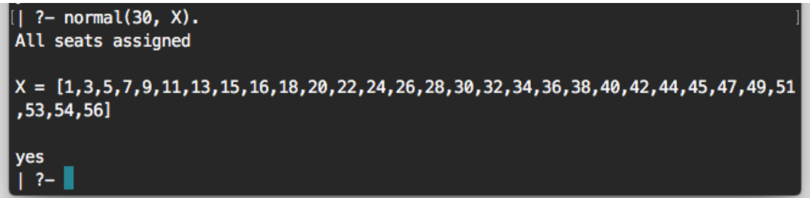


```
| ?- listcheckn([Adam,Brandon,Cullen], [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]).
All seats assigned

Adam = 1
Brandon = 3
Cullen = 5

yes
| ?-
```

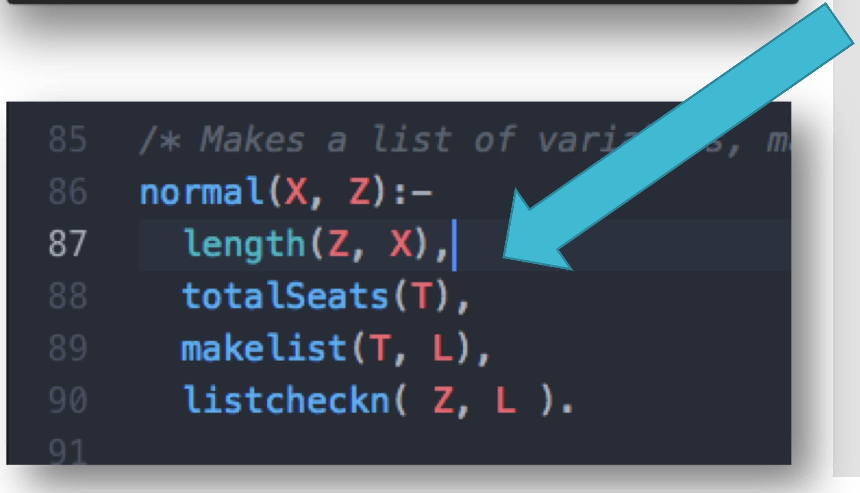
After



```
| ?- normal(30, X).
All seats assigned

X = [1,3,5,7,9,11,13,15,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,45,47,49,51
,53,54,56]

yes
| ?-
```



```
/* Makes a list of vari[illegible]s, m
normal(X, Z):-
  length(Z, X),
  totalSeats(T),
  makelist(T, L),
  listcheckn( Z, L ).
```