Forman Christian College, Lahore
(A Chartered University)

Assignment 5
Due: Monday 20/5/2013 at 12:01 am

Problem A: Generating a 2D Maze

Given a number n, write a function createMaze() that generates an n by n maze. In the maze, a wall is represented by the character '#', a place where someone can walk is represented by a space ' ' character, the starting point of the maze is marked by a lower-case 's', and the ending point of the maze is marked by a lower-case 'e'. There should be one starting 's' and ending point 'e' and multiple walls '#' and walkables ' '. Do not worry about how the maze comes out. In other words, it is alright if the end is not reachable from the start (this is left as bonus).

The size of the maze, n, is restricted as follows:

1.) n must be greater than or equal to 5.
2.) n must be odd.

It is your responsibility to ensure that the value of n is acceptable before calling the function.

Displaying the maze

After calling createMaze(), your array will contain n * n characters. Your job is now to print out the first n characters on one line, the second n characters on the next line, the third n characters on the next line, and so on, until all characters are displayed.

For example, suppose you have a maze where n = 5, and the following 25 characters are stored in the array after calling createMaze() (we have included numbers below each character to help make this more readable):

```
#####
s # e
# # #
#   #
#####
```

Output Sample

Here is one sample output of running the program. Note that this test is NOT a comprehensive test. You should test your program with different data than is shown here

based on the specifications given. The user input is given in *italics* while the program output is in bold.

Sample Run #1

```
What dimension shall we use for the maze? 2

Invalid input. Please enter an odd integer that is greater
than 4: -1

Invalid input. Please enter an odd integer that is greater
than 4: 4

Invalid input. Please enter an odd integer that is greater
than 4: 5

Here is the randomly generated maze:

#####
s   #
### #
#   e
#####
```

Sample Run #2

```
What dimension shall we use for the maze? 7

Here is the randomly generated maze:

#######
s     #
##### #
# #   #
# ### #
#     e
#######
```

Sample Run #3

```
What dimension shall we use for the maze? 7

Here is the randomly generated maze:

#######
s   # #
### # #
#   # e
### # #
```

```
#       #
#######
```

Problem B: ASCII Maze Game

Now modify your program from Problem A to allow the user to navigate the maze! As before, you will call createMaze() to populate your character array with an ASCII maze. This time, however, you will place a little person (represented by the '@' character) on the board, and allow him (or her) to move around the maze. You must prevent the character from stepping onto walls, and you must also detect when the character reaches the exit of the maze, and end the game.

Input Specification

1. n, where the size of the maze is n*n, will be an integer. **You** must perform the necessary checks to ensure that n is greater than or equal to 5, and that it is odd!
1. To navigate, the user will enter a single character and then hit enter. The user may enter a letter you're not expecting, in which case you must catch the error! **(Note: you MUST use the characters i, m, j, and k to represent moves up, down, left, and right, respectively. The user has an option to end the game by selecting 'q'. The TA would not want to have to adapt to new navigation controls in each program they grade!)**

Output Sample

Here are two sample outputs of running the program. Note that this test is NOT a comprehensive test. You should test your program with different data than is shown here based on the specifications given. The user input is given in *italics* while the program output is in bold.

Sample Run #1

What dimension shall we use for the maze? *3*

Invalid input. Please enter an odd integer that is greater than four: *5*

```
#####
@  #
# ###
#  e
#####
```

Make your move.

(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#####
s@ #
# ###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *i*

You can't move up!

```
#####
s@ #
# ###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *m*

You moved down!

```
#####
s  #
#@###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You can't move right!

```
#####
s  #
#@###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *j*

You can't move left!

```
#####
s  #
#@###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *i*

You moved up!

```
#####
s@ #
# ###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *j*

You moved to the left!

```
#####
@  #
# ###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#####
s@ #
# ###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *m*

You moved down!

```
#####
s  #
#@###
#  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *m*

You moved down!

```
#####
s  #
# ###
#@  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *m*

You can't move down!

```
#####
s  #
# ###
#@  e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#####
s  #
# ###
# @ e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#####
s   #
# ###
#  @e
#####
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#####
s   #
# ###
#   @
#####
```

You win!

Sample Run #2

What dimension shall we use for the maze? *7*

```
#######
#   # e
# ### #
# # # #
@ # # #
#     #
#######
```

Make your move.

(i = up, m = down, j = left, k = right, q = quit): *r*

That input is not valid. Please try again.

```
#######
#   # e
# ### #
# # # #
@ # # #
#     #
#######
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#######
#   # e
# ### #
# # # #
s@# # #
#     #
#######
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You can't move right!

```
#######
#   # e
# ### #
```

```
# # # #
s@# # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): i

You moved up!

#######
#   # e
# ### #
#@# # #
s # # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): i

You moved up!

#######
#   # e
#@### #
# # # #
s # # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): i
```

You moved up!

```
#######
#@  # e
# ### #
# # # #
s # # #
#     #
#######
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#######
# @ # e
# ### #
# # # #
s # # #
#     #
#######
```

Make your move.
(i = up, m = down, j = left, k = right, q = quit): *k*

You moved to the right!

```
#######
#  @# e
# ### #
# # # #
```

```
s # # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): m

You can't move down!

#######
#  @# e
# ### #
# # # #
s # # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): k

You can't move right!

#######
#  @# e
# ### #
# # # #
s # # #
#     #
#######

Make your move.
(i = up, m = down, j = left, k = right, q = quit): q
```

You quit before exiting the maze! Better luck next time.

Bonus

While generating the maze, we did not care that the end is reachable from the start. Can you modify your function, generateMaze() so that it generates a maze which can successfully end.

Submission instructions

You need to email the file, Maze.java to TA.comp102@gmail.com with the subject as "COMP 102: Assignment 5 - Roll number."

Notes on Grading

- All variables should have informative names.
- Your input/output should match the examples.
- Every program must have comments. There should be header comments in your program including your name and a brief introduction of the program. You should have at least one comment besides that.
- Any plagiarism/cheating would be strictly dealt with.
- Use correct indentation.
- 1 day late assignment would be penalized by a 10% grade deduction. 2 days late would be penalized by 25% grade deduction. Assignments submitted 2 days after the submission date would not be accepted.
- Failure to send Maze.java would result in a zero for the assignment.