1. (4 points) Square.toString()
   Here's what I would do for that. A switch statement is fine too:

```
public String toString() {
      if (type == EMPTY)
            return "_";
      else if (type == WALL)
            return "#";
      else if (type == START)
            return "S";
      else if (type == EXIT)
            return "E";
}
```

2. (4 points) Maze.toString()

```
public String toString() {
      String S = new String();
      for (int i = 0; i < Rows; i++) {
            for (int j = 0; j < Cols; j++)
                  S = S+maze[i][j].toString();
            S = S+"\n";
      }
      return S;
}
```

3. operations

Queue	addL(1)	1
	add(2)	1 2
	remove()	2
	add(3)	2 3
	add(4)	2 3 4
	remove()	3 4
	remove()	4
	add(5)	4 5

Stack							
add(1)	add(2)	remove()	add(3)	add(4)	remove()	remove()	add(5)
1	2 1	1	3 1	4 3 1	3 1	1	5 1

4. Maze algorithm

	at the start	after step 1	after step 2	after step 3	after step 4	after step 5	after step 6	after step 7	after step 8
worklist as a stack	(6,4)	(6,3) (6,5)	(6,2) (6,5)	(6,1) (6,5)	(6,0) (6,5)	(5,0) (6,5)	(4,0) (6,5)	(4,1) (3,0) (6,5)	(4,2) (3,0) (6,5)
newly explored square	N/A	(6,4)	(6,3)	(6,2)	(6,1)	(6,0)	(5,0)	(4,0)	(4,1)
newly marked square(s)	N/A	(6,3) (6,5)	(6,2)	(6,1)	(6,0)	(5,0)	(4,0)	(4,1) (3,0)	(4,2)

5. Printing the solution

```
make a stack of Squares
set p = exit square
while (p != start square) {
        push p onto the stack
        p = p.previous
}
push the start square onto the stack
while (stack is not empty) {
        print the stack top row and column
        pop the stack
}
```