Risk Roller

Professor Don Colton

Brigham Young University—Hawaii Campus

1 Overview

Risk (TM) is a board game sold by Parker Brothers, a division of Hasbro (TM). Wikipedia has a good article describing the game. If you are not familiar with Risk, you should read it.

`http://en.wikipedia.org/wiki/Risk_(game)`

In fact, you are encouraged to try playing the game.

`http://gamehouse.com/download-games/risk/`

This assignment is to write a program that rolls the dice to carry out attacks and defenses according to the rules of the game.

2 Battle

A battle consists of a series of attacks between two territories.

Each attack pits one, two, or three attacking armies against one or two defending armies.

To carry out the attack, roll (normal six-sided) dice according to the number of armies involved. For example, if there are two attacking armies and one defending army roll two dice for the attacker and one for the defender.

The highest-number attacking dice goes against the highest-number defending dice. If the attacker has a higher number, he wins. If the defender ties or has a higher number, the attacker loses.

After the attack, the loser has one army removed.

If both sides roll at least two dice, the process is repeated with the second-highest die from each side.

These attacks can continue as long as both attacker and defender have armies left.

3 Limitations

In each attack, the attacker can use up to three armies. However, the attacker is also limited to his total armies minus one. The minus one is because one army must remain out of combat so that when the turn is over it can defend against other attacks.

In each attack, the defender can use up to two armies. The defender is also limited to his total armies. All existing armies can participate in the defense.

4 Program Specifications

When your program first starts, it should clearly identify you as its author. You might say something like "Don's Fabulous Risk Roller." Or whatever. Just clearly identify yourself.

Your program should loop endlessly asking for inputs, performing the requested attacks, reporting the results, and starting over.

4.1 Inputs

On each battle your program must ask for three things:

(a) the number of attacker armies;

(b) the number of defender armies;

(c) the maximum number of attacks to carry out.

4.2 Repetition

After learning these three things, your program should go into a loop carrying out attacks as long

as possible. Stop when the attacker has only one army left. Stop when the defender has no armies left. Stop when the maximum number of attacks has been completed.

4.3 Reporting

Before the attacks begin, report how many armies each side has.

For each attack, report what the dice rolls were. Then report how many armies were lost by each side. Finally report how many armies each side has left.

5 Useful Code

Here are some pieces of code that might be useful in writing your program.

```
sub dice { return int ( rand(6) + 1 ) }
```

The dice subroutine will create a random number, one of 1, 2, 3, 4, 5, or 6, with equal likelihood. It simulates the rolling of a die.

```
@aDice = sort ( dice(), dice(), dice() );
```

This rolls the dice three times. The resulting numbers are sorted and stored in the array `@aDice`.

```
if ( $aDice[-1] > $dDice[-1] ) {
  $dMen-- } else { $aMen-- }
```

The last number in aDice (presumably the highest roll) is compared to the last number in dDice. If aDice is higher, dMen is reduced by one. Otherwise aMen is reduced by one.

```
sub combat {
  my ( $aMen, $dMen ) = @_;
  # carry out combat
  return ( $aMen, $dMen ) }
```

The combat subroutine receives two inputs from the `@_` array. It carries out combat (not shown here) and returns the number of men remaining.

```
($aMen,$dMen) = combat($aMen,$dMen);
```

This calls a combat subroutine, passing in the number of attackers (`$aMen`) and defenders. Upon return, the number of attackers and defenders is updated.

6 Online

You may be asked to port your Risk Roller program to the web. If so, it should act the same as before with a few important differences.

(a) Inputs will be from a web page instead of standard input. A separate handout will explain how to receive such inputs.

(b) When your program starts, it will create a web page that explains the game and has three blanks for the inputs.

(c) Each time your web page is submitted the browser, your program will start, extract the three inputs, conduct the battle, display the results, and stop. It will not be in an infinite loop, but it may seem like an infinite loop to the user.

(d) Graphics should be used to represent the dice rolls.

(e) Your web page should look wonderful. Don't forget to clearly identify yourself on the web page.

7 Useful Code

This might be useful in porting your program to the web.

```
print "<img src=$x.jpg alt=$x>\n";
```

If `$x` is a number between 1 and 6, then this will insert a jpg file into your web page, something between 1.jpg and 6.jpg. Or you could use `w$x.jpg` to get w1.jpg through w6.jpg.

```
print "<h1>Don's Risk Roller</h1>\n";
```

This will put a heading on your page.

```
print "<br/>\n";
```

This will break the current line and make your web page go to a new line.