

## Assignment 4

# Random Numbers and Modular Development

---

**Due: Wednesday, October 27**

In this assignment, your job is to write a program that simulates the operation of a slot machine of the sort you might find in a Nevada casino. The assignment is divided into two parts. In the first, your mission is to implement a simple text-based slot machine in which all of the output is displayed in the console window. In the second, your task is to extend the implementation so that the outcomes are displayed graphically as well. The first part depends only on the use of the Chapter 8 material on random numbers. In terms of how much work you will have to do, the second part is mostly a further exercise in using the graphics library described in Chapter 7. Part II, however, also requires you to decompose your program into two modules: one that simulates the slot machine and another that takes care of drawing the necessary figures on the graphics window. Although the idea of creating new interfaces is covered in detail in Chapter 8, it will probably also help you to read through the discussion of modular development in the introduction of Chapter 10.

Even though you only need to submit a single program that implements both parts of the assignment, make sure that you get Part I running before you move on to Part II.

### Part I

A typical slot machine has three wheels that spin around behind a narrow window. Each wheel is marked with the following symbols: **CHERRY**, **LEMON**, **ORANGE**, **PLUM**, **BELL**, and **BAR**. The window, however, allows you to see only one symbol on each wheel at a time. For example, the window might show the following configuration:



If you put a silver dollar into a slot machine and pull the handle on its side, the wheels spin around and eventually come to rest in some new configuration. If the configuration matches one of a set of winning patterns printed on the front of the slot machine, you get back some money. If not, you're out a dollar. The following table shows a typical set of winning patterns, along with their associated payoffs:

<b>BAR</b>	<b>BAR</b>	<b>BAR</b>	pays	\$250
<b>BELL</b>	<b>BELL</b>	<b>BELL/BAR</b>	pays	\$20
<b>PLUM</b>	<b>PLUM</b>	<b>PLUM/BAR</b>	pays	\$14
<b>ORANGE</b>	<b>ORANGE</b>	<b>ORANGE/BAR</b>	pays	\$10
<b>CHERRY</b>	<b>CHERRY</b>	<b>CHERRY</b>	pays	\$7
<b>CHERRY</b>	<b>CHERRY</b>	—	pays	\$5
<b>CHERRY</b>	—	—	pays	\$2

The notation **BELL/BAR** means that either a **BELL** or a **BAR** can appear in that position, and the dash means that any symbol at all can appear. Thus, getting a **CHERRY** in the first position is automatically good for two dollars, no matter what appears on the other wheels. Note that there is never any payoff for the **LEMON** symbol, even if you happen to line them up three of them.

Your task in Part I is to write a program that simulates playing a slot machine. Your program should provide the user with an initial stake of \$50 and then let the user play until either the money runs out or the user decides to quit. During each round, your program should take away a dollar, simulate the spinning of the wheels, evaluate the result, and pay the user any appropriate winnings. For example, a user might be lucky enough to see the following sample run:

```
Would you like instructions? no↵
You have $50. Would you like to play? yes↵
PLUM LEMON LEMON -- You lose
You have $49. Would you like to play? yes↵
PLUM BAR LEMON -- You lose
You have $48. Would you like to play? yes↵
BELL LEMON ORANGE -- You lose
You have $47. Would you like to play? yes↵
CHERRY CHERRY ORANGE -- You win $5
You have $51. Would you like to play? yes↵
LEMON ORANGE BAR -- You lose
You have $50. Would you like to play? yes↵
PLUM BELL PLUM -- You lose
You have $49. Would you like to play? yes↵
BELL BELL BELL -- You win $20
You have $68. Would you like to play? yes↵
ORANGE PLUM BELL -- You lose
You have $67. Would you like to play? yes↵
BAR BAR BAR -- You win $250
You have $316. Would you like to play? no↵
```

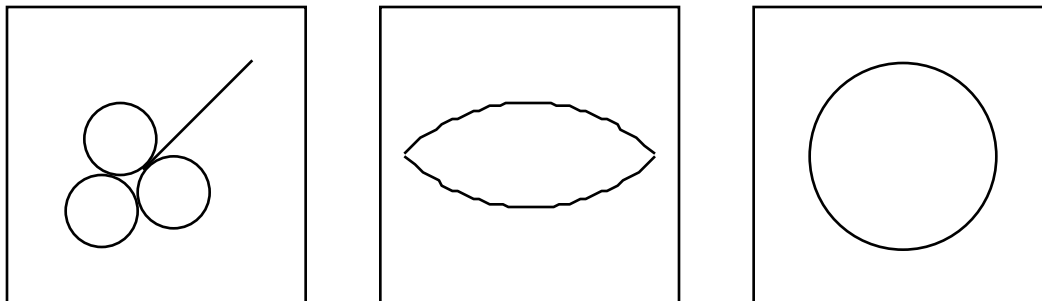
Even though doing so is not realistic (and would make the slot machine unprofitable for the casino), you should assume that each of the six symbols is equally likely on each wheel.

### Part II

Once you finish Part I, your next challenge is to add graphics to the program by drawing each of the symbols in the drawing window. For example, if your program produced the sequence

CHERRY LEMON ORANGE

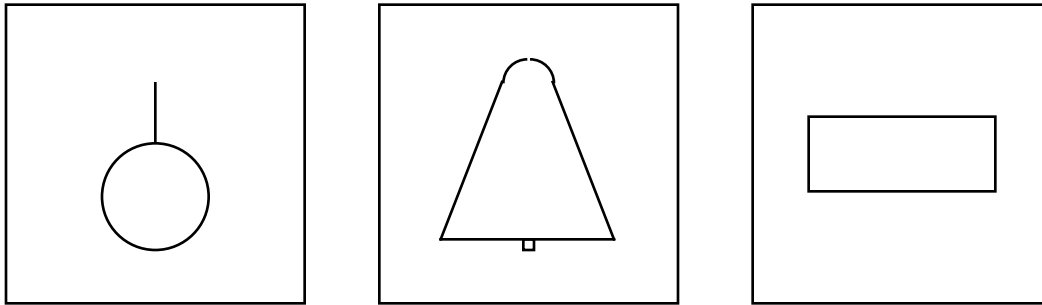
your program should create a display that looks something like this:



If the sequence were instead

PLUM BELL BAR

the output would appear as follows:



Unlike in the last assignment, which included a complete specification of all the constants necessary to draw these diagrams, part of your job is to work out what the elements of these figures are and how to draw them in roughly the right proportions.

The most important aspect of Part II of this assignment, however, is not the graphics but the overall organization. To meet the requirements of the assignment, the functions that draw the graphics on the screen must be part of a separate module. For example, if your main program module is in a file called `slots.c`, you need to put the graphical parts of the assignment in a different file, which might be `gslots.c`. Moreover, you need to create an interface called `gslots.h` that includes the definitions that are common to both sides. In splitting the code, all work with the graphics library—with the exception of the call to `InitGraphics`, which must be the first statement in the main program—must be part of the `gslots.c` implementation. A major part of the assignment is the nature of this modular decomposition, and you should make sure that the interface you design meets the criteria outlined in Chapter 8 as well as you can.

Note that each time you display a new set of symbols, you need to clear the screen first. The easiest way to clear the screen is to call `InitGraphics` again, which restores the graphics window to its initial empty state.

### Extending the assignment

Like Assignment #3, the slot machine offers lots of opportunities to try for + and ++ scores. Some of the ideas you might consider are:

- The drawings shown earlier in the handout are minimalist renderings of the symbols that appear on the wheels of a slot machine. If you finish the assignment early and want to get more ambitious, you can jazz them up a bit. For example, if you use the extended graphics library, you can draw the symbols in color and fill in the open spaces. If, however, you are just trying to meet the requirements of the assignment, you should draw figures that resemble the ones shown here as closely as you can.
- The program as designed offers very little suspense. In a real slot machine, the first wheel stops first, then the second, then the third, so that you have a chance to anticipate your winnings.
- A casino using your slot machine would quickly go broke because large payoffs happen too often. In a real slot machine, the chance of getting a **BAR** is not 1/6 on every wheel. Instead, each wheel might have 20 positions, only one of which is a **BAR** and three of which are **LEMONS**. How could you change your program to make it simulate more closely the actions of a real machine?