PGSS – Programming Lab
Task 4 Defining and Calling Your Own Functions, User Input, Scoring, and Timing

Class Notes and Code selections 04, 07, and 08 will be very helpful to you in this set of tasks.

You will continue working with the collision detection between your bouncing and wrapping figures. One part of today's task is to "clean" up your code by moving code segments into functions that are called by draw( ) into functions that you define. These new functions will be called from draw( ). In the second part you are going to add user (player) control the wrapping figure with either keyboard input or mouse input. You will also add scoring and timing of the game. The exact rules are up to you with one exception. The wrapping figure is being controlled by the user. If the user lets the wrapping figure go off screen, the user's score is set back to zero.

____ Start with either Task3 and make a copy of it named Task4.

Currently you have all of the code that is involved in animation jammed into draw( ) and it is getting messy and difficult to read. In Jim's world of happy programmers, he prefers (make that DEMANDS) that draw( ) be coded as the table of contents to the entire program. When you are done, your draw( ) function should resemble the following:

```
void draw( )
{
  prepScreen( );
  moveBouncer( );
  drawBouncer( );
  moveWrapper( );
  drawWrappper( );
  checkForCollission( );
}
```

Coded like this it is very easy for another programmer to quickly see what your program does. It is very easy to find code that has an error or needs to be modified.

____ Conceptually break your code into it specific, tasks. Define function for each task and move the appropriate code into the function. The order of these functions is not important but it is easier for many novices to code the function right under draw( ). Code one at a time and test it. Fix the errors before coding the next one. Code it right after draw( ). This saves scrolling back and forth between draw( ) and your new function. For this task, the argument lists for your functions (the stuff in the parentheses) will always be empty. More on the arguments of functions next week.

From an execution point of view, your newly organized program should run no differently that the original form of the program.

Now on to the control. The user will control the movement of the wrapping figure with either the mouse or the keyboard. The choice is up to you and both will be demoed in class. If you choose to let the user control the wrapper with the mouse, the user must not just drag the mouse to a collision. The mouse must "influence" the motion of the wrapper. Direct location of the wrapper with the mouse is not acceptable. This also will be demoed in class.

____ Add movement control for left, right, up and down movement. Wrapping must be allowed from to o bottom as well as bottom to top – this new. Same for left and right. Wrapping must be allowed to occur from left to right AND from right to left. This is also new and will take some serious thinking.

____ Once the wrapping control is working, add a scoring variable and track the user's prowess in driving the wrapper to a collision.

____ Once the scoring is done add a timer to the game. Give the user 30 seconds to play the game.