*Laboratory 4*

1. #The following procedure simulates montyhall problem.
   # See the attached NYTIMES front page article.
   # Run this to test whether it pays off to switch the doors.
   Tabulate your results, for the number of runs =10,20,30,40,50,60,70,80,90,1000.
   Can you give a simple explanation?

```
##########################################################################
#                                                                        #
#  COMMAND FILE:  PARADE.MAP (Version 1.0) for Maple V4.3 or later       #
#  DATE WRITTEN:  February 20, 1991                                      #
#  AUTHOR:        Frederick W. Chapman (FC03@NS.CC.LEHIGH.EDU)           #
#                                                                        #
##########################################################################
#                                                                        #
#  DESCRIPTION AND INSTRUCTIONS FOR USE:                                 #
#                                                                        #
#  This Maple command file simulates the Monty Hall game show problem    #
#  as described in Ms. Savant's column in Parade Magazine.  It has been  #
#  tested with Maple V4.3 for VAX/VMS systems and with Maple V5.0 for    #
#  386 DOS systems.  If the command file is stored in a file called      #
#  PARADE.MAP, the command file can be executed by simply entering       #
#                                                                        #
#      read `parade.map`;                                                #
#                                                                        #
#  at the Maple prompt (note the use of *backwards* single quotes to     #
#  delimit the file name).  The number of games played in the simulation #
#  is set by a variable called 'num_games'; the default value is 1000    #
#  games (which takes under 15 seconds to run on a 25 MHz 386 DOS        #
#  machine with an 80387 math coprocessor).                              #
#                                                                        #
##########################################################################


# define procedure to generate random integer from choices {1, 2, 3}
three := rand(1..3):

# define procedure to generate random integer from choices {1, 2}
```

```
two := rand(1..2):



montyhall := proc(num_games)
# initialize counters
win_via_keep := 0:
win_via_switch := 0:

##########  BEGIN LOOP  ##########

for game_count from 1 to num_games do

  # shuffle one prize and two goats behind doors 1, 2, and 3
  prize := three();

  # contestant makes first choice of three doors
  c1 := three();

  # host reveals a goat behind one of the doors NOT chosen
  if c1 = prize
    then
      # select one of two goats at random
      choices := {1, 2, 3} minus {prize};
      host := choices[two()];
    else
      # must select the other goat
      choices := {1, 2, 3} minus {prize, c1};
      host := choices[1];
    fi;

  # contestant elects to switch
  choices := {1, 2, 3} minus {c1, host};
  c2 := choices[1];

  # tabulate wins according to strategy
  if c1 = prize
    then
      win_via_keep := win_via_keep + 1;
    fi;
```

```
  if c2 = prize
    then
      win_via_switch := win_via_switch + 1;
    fi;

########## END LOOP ##########

od:

# compute empirical probabilities
prob_win_via_keep := evalf(win_via_keep/num_games);
prob_win_via_switch := evalf(win_via_switch/num_games);
print('prob_win_via_keep',prob_win_via_keep,' prob_win_via_switch',prob_win_via_switch);
# total probabilities should be one
total_prob := prob_win_via_keep + prob_win_via_switch;
end;
```

2. You modify the above program to do the calculations when there are four doors instead of three doors. Repeat the simulation to determine whether it pays off to switch?

3. Find the area of the circle from Lab 3.

4. Find the area of the ellipse from Lab 3.