An important element of engineering design is the use of simulation models to predict the behavior of the system under consideration. Physical models can be used (either full, pilot, or bench scale), but they are frequently expensive to build and time consuming to test. Computer based mathematical models provide a convenient and time efficient alternative to physical models for many design problems.

Monte Carlo simulation is the process of repeated simulation to observe the probability (relative frequency of likelihood) or behavior of a random event. In this lab exercise, Monte Carlo simulation will be used with a simple computer model to determine the relative advantage of one navigational strategy over another. Suppose that two children (Jack and Jill) were dropped off at a movie theater one Saturday morning with instructions to walk home after the movie is over. When leaving the movie, their brains are so mushed, they seem to have forgotten how to get home. They do know that their house and the theater are within a grid of streets bounded on all four sides by highways.

Jack and Jill are unable to agree on a method to find their home, so they set out on separate paths. Jack decides that he will start in a random direction until he gets to an intersection. At the intersection, he will choose which direction to continue in a random fashion (including heading back in the same direction he just came). Jill will use what she feels is an improved version of Jack’s approach. When she reaches an intersection, she will choose a random new direction, but will not ever backtrack in the direction she just came from. Neither child will cross a highway.

Write a program that will allow you to compare the two navigational strategies. Assume that the grid of streets is 15 by 15, with the highway border on rows 1 and 15, and column 1 and 15. Allow the user of the program to input the location of the theater and the house, along with a flag specifying which navigational strategy should be used. For each set of input data, have the program run through 50,000 attempts at getting home, and compute the average number of steps (intersections reached) required to get home. This program MUST contain at least one function subprogram and one subroutine. Turn in your results for starting and ending positions (2, 2) and (14, 14).

To choose a new direction to travel at each intersection, we need to be able to get uniformly distributed random integers in the range 1 to 4. The computer can pick random numbers uniformly distributed on a specified interval using a pseudo random number generator. To generate random real numbers between 0.0 and 1.0, call the intrinsic subroutine \texttt{random number}, i.e.

\begin{verbatim}
call random\_number(x)
\end{verbatim}

where the argument x will be the random real number. Now to get a random integer between LL and UL, use the statement

\begin{verbatim}
i=int(x*(UL-LL+1))+LL
\end{verbatim}

where i will be the random integer. Before the first use, the random number generator should be initialized with a call to the intrinsic \texttt{random\_seed} subroutine. The exact method used to initialized the random number generator so that it generates a new sequence of values for each run of the program differs from one compiler to another. The subroutine below (from the gfortran users manual) has been designed to work with any Fortran 95 compiler, and I encourage you to use it (this does NOT count as one as your required subroutine!).
subroutine init_random_seed()
  integer::i,n,clock
  integer, dimension(:), allocatable :: seed
  !Randomly initialize the random number generator
  call random_seed(size=n)
  allocate(seed(n))
  call system_clock(count=clock)
  seed = clock + 37 * (/ (i - 1, i = 1, n) /)
  call random_seed(put=seed)
end subroutine init_random_seed