Monty Hall, the game show host, opens one of the other two doors (doors 2 or 3) to reveal a goat.
You now have two choices:
- **stay** -- stay with your first choice and open door 1
- **switch** -- switch to the other closed door.
You win if the door you open has the car.

**Classwork 28.1(7) montyhall.f95**
First write a function `cardoor()` which randomly puts a card behind one of the three doors: doors 1, 2, 3
Write a function `winningchoice()` which shows which choice won: **stay** if the car is behind the door 1, **switch** if it is behind the other closed door. Run this a million times to find which choice wins most often.

**Monty Hall Problem**
There are three closed doors: doors 1, 2, 3.
A car is hidden behind a randomly chosen door, call it `cardoor ∈ {1, 2, 3}`. The other two doors have goats.
You choose one of the doors, call it `firstchoice`. Since they all have the same probability of having a car, let’s suppose `firstchoice=1`.

---

**190 Lecture 28 Probability**
Final: Thursday 12:00-2:00, here in PSB 208, sit 2 apart
Mathematical modeling and simulation.
`rand()` gives randomly generated numbers in \([0, 1)\).
To find the probability of an event \(X\) (say “heads”) when running a process \(P\) (say tossing a coin):
Run the process many times, say `nruns= inf=10**6`.
Count the number, `numx`, of times event \(X\) occurs.
Approximately, the probability of \(X=\frac{numx}{nruns}\).
But, instead, you must write \(X=\frac{real(numx)}{nruns}\) since Fortran divides reals, not integers.
Correct this program. Should get 0.5, not 0.
character(6) function winningchoice()
integer::cardoor, firstchoice=1
if(cardoor()==firstchoice) then
    winningchoice = ____  ! 'stay' or 'switch'?
else;  winningchoice = ____  ! 'stay' or 'switch'?
endif !return which is the winning choice, '...' need the quotes.
endfunction

program monty_hall
character(6)::winningchoice
integer::nstay,nswitch,nruns=10**6
real::probstay,probswitch
nstay = _______  ! nstay = # of times staying wins, start count
nswitch = _______  ! nswitch = # of times switching wins
    do i=1,nruns
        if(winningchoice()=='stay') then
            nstay = _______  
        else; nswitch = _______  
    endif
endo
do! probstay = probability staying wins
    probstay = ______________  
! probswitch= probability switching wins
    probswitch = ______________  
print *,'Prob. staying wins:',probstay
print *,'Prob. switching wins:',probswitch
endprogram ! 

If cardoor() picks doors randomly, the doors should be about equally likely.

Delete the cardoor_test program and paste in the next function and program.
**Random Integers**

\( r = \text{rand}() \) is a random real in \([0, 1)\).

\( 4r \) is a random real in \([0, 4)\).

\( \floor{4r} \) rounds these down to random integers \(0, 1, 2, 3\).

\( \floor{(n+1)r} \) rounds gives random integers \(0, 1, 2, 3, \ldots, n\).

\( m + \floor{(n+1-m)r} \) gives random integers \(m, m+1, \ldots, n\).

Let \( \text{randint}(m, n) \) be these random integers from \(m\) to \(n\).

### randint.f95

```fortran
integer function randint(m, n)
  integer:: n, m, values(8)
  character(10):: date, time, zone
  call date_and_time(date, time, zone, values)
  randint = m + mod(values(8), n+1-m)
endfunction
```

### randint_test.f95

```fortran
program randint_test
  integer:: randint
  character(:):: a
  do
    print *, "Press any key. Press 0 to quit."
    read*, a; if(a=="0") exit
    print *, randint(4,7)
  enddo
endprogram
```
Suppose you want to randomly pick integers from 1, 2, 3, 4 but instead of their being equally likely, you want their probabilities to be prob = [prob(1), prob(2), prob(3), prob(4)] = [.2, .2, 0, .6], then use the following randint_prob(prob, n) function.

```plaintext
!randint_prob.f95
integer function randint_prob(prob, n)
real::prob(n), r, p
! picks numbers from 1, ..., n, i has probability prob(i)
r=rand(); p=0
do i=1,n
  if(p<r .and. r<p+prob(i) )then
    randint_prob=i; return
  endif
  p=p+prob(i)
d enddo
endfunction
```

```plaintext
program test_rand_choose
real::prob(4)
integer::i, randint_prob
prob=(/ .2, .2, 0., .6 /)
do i=1,20;
  print*, randint_prob(prob, 4)
d enddo
endprogram
```

---

Given a deck of n cards, cards=[1,2,3,...,n], you must shuffle the deck, i.e., randomly permute the cards. If n=52, the shuffled deck might look like cards=[28,49,33,8,...,6].

Algorithm: for each i from 1 to n, use randint(i, n) to randomly pick a j in {i, i+1, ..., n}. Then swap cards in positions i and j, i.e., swap cards(i) and cards(j).

```plaintext
!c28_2_4shuffle.f95
integer function randint(m, n)
  randint = m+floor((n+1-m)*rand())
endfunction
subroutine swap(i, j)
  k=i; i=j; j=k;
endsubroutine
subroutine shuffle(cards, n)
integer::cards(n), randint
... replace with lines needed to shuffle a deck of n cards
endsubroutine
program randint_test
integer::cards(52), randint
do i=1,52; cards(i)=i; enddo
print*, 'First 13 cards of 5 shuffles.'
do i=1,5
  call shuffle(cards, 52)
  print ' (13(i3), a)', (cards(j), j=1,13)," ..."
d enddo
endprogram
```
Write a program which plays rock-paper-scissors. For the benefit of online, we use date_and_time for random values. youmove is the move the user picks, mymove is the move the computer randomly generates. Use randint to randomly pick 1, 2, or 3 where 1 = rock, 2 = paper, 3 = scissors. winner(i, j) = i if i wins, j if j wins, 0 on a draw.

```fortran
!h28_1_3rockpaperscissors.f95
integer function randint(m,n)
integer::n,m,values(8)
character(10)::date,time,zone
call date_and_time(date,time,zone,values)
randint = m+mod(values(8),n+1-m)
endfunction

program rockpaperscissors
integer::youmove,mymove,randint
integer::win,winner(3,3)
integer::rock=1,paper=2,scissors=3,draw=0
character(3)::move(3)
moves(1)= 'rock'; moves(2)= 'paper'; moves(3)= 'scissors'
winner(rock,:) = [draw,paper,rock]
winner(paper,:) = [paper,draw,rock]
winner(scissors,:) = __________
print*, 'Enter 0 to quit.'
do
```

Suppose data is coming in a stream, 4, 2, 3, 2, 6, .... And, we wish to keep a running average. As each new data item comes in, we need a way to update the average without retaking the total every time. Suppose the first \( n \) items are \( a_1, a_2, a_3, ..., a_{n-1}, a_n \). Suppose \( A_{n-1}, A_n \) are the respective averages of the first \( n - 1 \) and first \( n \) items. Then

\[
A_{n-1}(n-1) + a_n)/n
\]

\[
= (a_1 + a_2 + a_3 + ... + a_{n-1} + a_n)/n
\]

\[
= a_1 + a_2 + a_3 + ... + a_{n-1} + a_n)/n = A_n
\]
A drunk starts at the origin \([0,0]\). Each step is randomly 1 unit to the right, 1 unit up, 1 unit to the left or 1 unit down. On average, how far from the origin will he after 100 steps?

```fortran
integer function randint(m,n)
    randint = m+floor((n+1-m)*rand())
endfunction

subroutine onestep(s)
    integer::s(2),i,randint
    i = ______  !use randint to pick i randomly from 1,2,3,4
    if(i==1) then; s=[1,0]  !right
    elseif(i==2)then; s=[0,1]  !up
    elseif(i==3)then; s= ______  !left
    else; s= ______  !down
    endif
end subroutine

subroutine randomwalk(n,w)
    integer::s(2),w(2)
    w = ______  !random walks start at the origin
    do i=1,n;
       ______  !use onestep to generate a random step
       w = ______  !update the position by adding step s.
    enddo
end subroutine

real function distance(w)
    integer::w(2)
    distance=real(w(1))**2+real(w(2))**2
endfunction
```

```fortran
real function average_dist(n)
!average distance from origin after n random steps
real::distance
integer::w(2),inf=10**3
do i=1,inf
    call randomwalk(n,w)
    ______  ! update the running average
enddo
end function
```

```fortran
program average_walk_distance
real::average_dist
do n=10,100,10
    print*,'Distance after',n,'steps:',average_dist(n)
enddo
endprogram
```

Distance after 100 steps rounds to 9.