Pseudokod

# Omega

integer n tärningskast

integer m sidor

integer[][] Omega( n, m )

{

integer[][] omega

integer i, j

for( 0 < i < m^n)

{

for( 0 < j < n )

{

omega[i][j] = (i/( m^j ) %m )+1

}

}

return omega

}

# E

integer[][] E( n, m )

{

integer[][] e

integer i, die

boolean allValues, set to True

for( 0 < i < omega.size )

{

for( 1 < die <= m )

{

if ( omega[i] does not contain die )

{

allValues = false

}

}

if( allValues = true )

{

put omega[i] in e

}

}

return e

}

# F

integer[][] F( n, m )

{

integer[][] f

integer i, j

boolean lessThan, set to False

for ( 0 < i < omega.size )

{

for ( 1 < j < omega[i].size )

{

if ( omega[i][j] < omega[i][j-1] )

{

lessThan = true

}

}

if ( lessThan = false )

{

put omega[i] in f

}

}

return f

}

# Section

integer[][] A, B

integer[][] section( A, B )

{

integer[][] sect

integer i, j

for ( 0 < i < A.size )

{

for ( 0 < j < B.size )

{

if ( A[i] == B[i] )

{

put A[i] in sect

}

}

}

return sect

}

# Probability

Probability()

{

float pE = ( e.size / omega.size )

float pF = ( f.size / omega.size )

float pE|F = ( sect.size / f.size )

printeger “P(E):” pE

printeger “P(F):” pF

printeger “P(E|F):” pEF

}

# Kasta\_tarning

integer kasta\_tarning( m )

{

if ( m < 2 )

{

return NULL

}

integer value = randomValue % m

return value

}

# Kasta\_tarningar

integer n, m

integer[] kasta\_tarningar( n, m )

{

integer[] values

for ( 0 < i < n )

{

put kasta\_tarning( m ) in values

}

return values

}

# 1b Probability

integer n, m, x

oneBprobability( n, m, x )

{

integer[][] values

for ( 0 < i < x )

{

int[] tempValues = kasta\_tarningar( n, m )

put tempValues in values

}

omega = values

print “E: ” E( n, m ).size

integer b = F( n, m ).size

print "F: " b

integer a = section( E( n, m ), F( n, m ).size )

if (b ≠ 0)

{

float probability = a / b

print a

print b

print n m “-sided dice thrown” x “times gives P(E|F): ” probability

}

}

# Binom\_pascal

integer n, k

integer binom\_pascal( n, k )

{

if ( k > n )

{

return 0

}

integer[][] pascalVector

for ( 0 < i <= n ) // Row number

{

integer[] rowVector

for ( 0 < j <= i ) // Element in row

{

if ( j = 0 or j = i )

{

put 1 in rowVector

print 1

}

else

{

integer firstValue = pascalVector[i – 1] [j – 1]

integer secondValue = pascalVector[i - 1][j]

print (firstValue + secondValue)

put (firstValue + secondValue) in rowVector

}

}

put rowVector in pascalVector

}

return pascalVector[n][k]

}

# Expansion

string x, y

integer n

expansion( x, y, n )

{

for ( 0 < i <= n )

{

integer k = binom\_pascal( n, i )

if ( k > 1 )

{

print k

}

if ( ( n – i ) ≠ 1 )

{

if ( ( n – i ) ≠ 0 )

{

print "(" x ")^" (n – i)

}

}

else

{

print x

}

if ( i ≠ 1 )

{

if (i ≠ 0)

{

print " (" y ")^" i

}

}

else

{

print y

}

if (i ≠ n)

{

print " + "

}

}

}

# Kasta\_mynt

float p

integer kasta\_mynt( p )

{

integer probability = p \* 1000

integer result = randomValue % 1000

if (result < probability)

{

return 1

}

else

{

return 0

}

}

# Mynt\_experiment

integer n, k

float p

boolean mynt\_experiment( n, k, p )

{

integer count = 0

for ( 0 < i < n )

{

count = ( count + kasta\_mynt( p ) )

}

return true if ( count = k )

}

void Coin::print\_experiment( int n, float p)

{

int x = 100000000

for (int k = 0 k <= n k++)

{

int h = 0

for (int i = 0 i < x i++)

{

if (experiment( n, k, p))

{

h++

}

}

cout << "k: " << k << " h/x: " << (float)h/(float)x << endl

}

}