```
> #Hrudai Batini Hw7

  read "/Users/hb334/Documents/M7.txt";
  with(Statistics):
  with(LinearAlgebra);
  Help7();
```

[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CARE, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, CompressedSparseForm, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, FromCompressedSparseForm, FromSplitForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct, LA_Main, LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, ProjectionMatrix, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]

GR(p,i,N), GRt(p,i,N), GRm(N,p), OneStepMarkov(P,i), MarkovTrip(P,K), StSa(P,K) , StS(P), StSp(P,K), RandSM(N) **(1)**

```
> #1
  EstGR := proc(p,i,N,K)
  local c,prob,r,cle;
  r := 0;
  c := 0;
  prob := 0;
  while r<K do
  cle := GRt(p,i,N);
  prob := prob + cle[1];
  c := c + cle[2];
  r := r+1;
  end do;
  prob := prob/K;
  c := c/K;
  print('ProbabilityOfExitingAWinner',prob); print
  ('AverageDurationOfGame',c); RETURN([prob,c]);
```

```
end proc;
EstGR(0.5,5,10,100);
```

$EstGR := \mathbf{proc}(p, i, N, K)$

```
    local c, prob, r, cle;
    r := 0;
    c := 0;
    prob := 0;
    while r < K do
        cle := GRt(p, i, N); prob := prob + cle[1]; c := c + cle[2]; r := r + 1
    end do;
    prob := prob/K;
    c := c/K;
    print('ProbabilityOfExitingAWinner', prob);
    print('AverageDurationOfGame', c);
    RETURN([prob, c])
end proc
```

$$ProbabilityOfExitingAWinner, \frac{23}{50}$$

$$AverageDurationOfGame, \frac{627}{25}$$

$$\left[\frac{23}{50}, \frac{627}{25}\right] \quad \textbf{(2)}$$

```
> #2
#A) The given linear homogeons equation is true becuase it takes
into account the probablity of a fair coin, 1/2 being equal for
either a head or tails. The possibility of losing is just as
likely as the possibilty of winning and the boundary conditions
are the possibilties of achieving the goal at N and losing at 0.
To infinity the probablity of this problem is 1/2 as the
reccurence conditions are only 0 and 1.
#B) The closed form expression yN(i) = i/N satisfies the
recurrence as plugging it into the reccurence returns i/N for all
values of i. The value of 0 for i would return 0 for the
probabilty and N for i would return 1. Hence the losing and
winning conditions are taken into account and the explicit
formula xN(i) = i/N can be established.
#C) The linear reccurence is true for deptermingin the number of
rounds as this specific problem is simplified to the number of
coin flips equates to the number of rounds. The boundary
conditions of 0 and N are set to 0 becuase atleast 1 flip is
required at the most minimal conditon for this problem and that
constraint is accounted for in the equation with the +1 outside
the 1/2(...).
#D) Plugging in zN(i) = i(N-i) into the recurrence returns, i(N-
i). The boundary conditons are the same as when i=0 and N; the
value returns to 0. Hence the reccurence is equal to EN(i) = i(N-
i).
```

```
> #3
ExactFairGR:= proc(i,N)
```

```
  local x, c, XN,d;
  x:=i;
  c:=0;
  while 0<x and x<N do
  XN := RandomVariable(Bernoulli(i/N));
  c:= c+1;
   d:=trunc(Sample(XN,1)[1]);
  if d=0 then
      x:=x-1;
  else
  x:=x+1;
  end if;
  end do;
  RETURN([i/N,c]);
  end proc;
  x:= 1;
  while x<=19 do
  ExactFairGR(x,20);
  EstGR(1/2,x,20,3000);
  end do;
```

$ExactFairGR := \textbf{proc}(i, N)$

local x, c, XN, d;

$x := i$;

$c := 0$;

while $0 < x$ **and** $x < N$ **do**

$XN := Statistics\text{:-}RandomVariable(Bernoulli(i/N))$;

$c := c + 1$;

$d := \mathrm{trunc}(Statistics\text{:-}Sample(XN, 1)[1])$;

if $d = 0$ **then** $x := x - 1$ **else** $x := x + 1$ **end if**

end do;

$RETURN([i/N, c])$

end proc

$$x := 1 \tag{3}$$

```
> #4 #k=4000 was causing the software to crash repeatededly.
  p := RandSM(10):
  StSa(p,400);
  evalf(StSp(p,400));
  evalf(StS(p));
```

[0.1075000000, 0.08500000000, 0.1225000000, 0.09750000000, 0.1150000000, 0.08750000000, 0.1000000000, 0.1050000000, 0.1025000000, 0.07750000000]

[0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510, 0.1043624510]

[0.1043624510, 0.09411070242, 0.1003014246, 0.1163199561, 0.1080399967, 0.08689788956, 0.09454729287, 0.09368425315, 0.09873286874, 0.1030031649] (4)

```
> #Hrudai Battini Problem 6 Hw7
  with(Statistics):
> # Optional 6
  EstimateProbSum := proc(p1,p2,p3,p4,p5,p6,N1,N2,K1,K2)
  local k1,k2,a,b,c,d,e,f,p,x,ps;
  a:=0;
  b:=0;
  c:=0;
  d:=0;
  e:=0;
  f:=0;
  ps:=0;
  k2:= 0;
  p :=0;
  while(k2<K2) do
  x:=0;
  k1:=0;
  while(k1<K1) do
  p:= rand(0.0..1.0);
  if(p()<p1) then a:= 1+a;
  elif(p()>p1) and (p()<=(p2+p1)) then b:= b+2;
  elif(p()>(p1+p2)) and (p()<=(p3+p2+p1)) then c:= c+3;
  elif(p()>(p1+p2+p3)) and (p()<=(p4+p3+p2+p1)) then d:= d+4;
  elif(p()>(p1+p2+p3+p4)) and (p()<=(p5+p4+p3+p2+p1)) then e:= e+5;
  elif(p()=1) or (p()>(p5+p4+p3+p2+p1)) then f:= f+6;
  end if;
  k1 := k1 + 1;
  end do;
  if(a>=N1) and (a<=N2) then x:= 1;
  elif(b>=N1) and (b<=N2) then x:=1;
  elif(c>=N1) and (c<=N2) then x:=1;
  elif(d>=N1) and (d<=N2) then x:=1;
  elif(e>=N1) and (e<=N2) then x:=1;
  elif(f>=N1) and (f<=N2) then x:=1;
  end if;
  if(x=1) then ps:= ps+1; end if;

  k2 :=k2+1;
  end do;
  print(ps);
  RETURN(ps/K2);
  end proc;
```

$EstimateProbSum := \mathbf{proc}(p1, p2, p3, p4, p5, p6, N1, N2, K1, K2)$ **(1)**

$\quad \mathbf{local}\ k1, k2, a, b, c, d, e, f, p, x, ps;$

$\quad a := 0;$

$\quad b := 0;$

$\quad c := 0;$

$\quad d := 0;$

$\quad e := 0;$

$\quad f := 0;$

$\quad ps := 0;$

$\quad k2 := 0;$

```
    p := 0;
    while k2 < K2 do
        x := 0;
        k1 := 0;
        while k1 < K1 do
            p := rand(0...1.0);
            if p( ) < p1 then
                a := a+1
            elif p1 < p( ) and p( ) <=p2+p1 then
                b := b+2
            elif p2+p1 < p( ) and p( ) <=p3+p2+p1 then
                c := c+3
            elif p3+p2+p1 < p( ) and p( ) <=p4+p3+p2+p1 then
                d := 4+d
            elif p4+p3+p2+p1 < p( ) and p( ) <=p5+p4+p3+p2+p1 then
                e := e+5
            elif p( ) =1 or p5+p4+p3+p2+p1 < p( ) then
                f := f+6
            end if;
            k1 := k1+1
        end do;
        if N1 <=a and a <=N2 then
            x := 1
        elif N1 <=b and b <=N2 then
            x := 1
        elif N1 <=c and c <=N2 then
            x := 1
        elif N1 <=d and d <=N2 then
            x := 1
        elif N1 <=e and e <=N2 then
            x := 1
        elif N1 <=f and f <=N2 then
            x := 1
        end if;
        if x=1 then ps := ps+1 end if;
        k2 := k2+1
    end do;
    RETURN(ps/K2)
end proc
> EstimateProbSum(1/6,1/6,1/6,1/6,1/6,1/6,100,330,360,1000);
```

$$\frac{1}{200} \tag{2}$$

```
> EstimateProbSum(0.1,0.1,0.1,0.1,0.1,0.5,100,430,470,1000);
```

$$\frac{9}{1000} \tag{3}$$

```
>
```