TI82/TI83/TI84 Calculator Mathematical Programs

These calculator programs have been written by James Jones to aid his students. Other people may find the programs useful and the source code is provided below in case people want to enter them into their calculators.

Downloads of the binary versions of the program are available at https://www.richland.edu/james/ti82/

James Jones
Professor of Mathematics
Richland Community College
One College Park
Decatur, IL 62521-8512
(217) 875-7211, ext. 490
james@richland.edu

Programs

ALGEBRA

General algebra routines. Will solve a quadratic equation, return the greatest common factor of a list, factor a quadratic, perform synthetic division, or solve a system of equations using Cramer's rule.

```
: Menu("ALGEBRA","QUADRATIC",1,"GCF",2,"FACTOR",3,"S
  YNTHETIC DIV",4,"CRAMERS RULE",5,"QUIT",6)
: Lbl 1
: Menu("QUADRATIC SOLVER","COEFFICIENTS",A,"SOLVE
  Y₁",B,"SOLVE Y₂",C
: Lbl A
: Prompt A
: Prompt B
: Prompt C
: Goto D
: Lbl B
: Y₁(Ø)→C
: Y₁(1)-C→D
: ((Y₁(⁻1)-C)+D)/2→A
: Y₁(1)-A-C→B
: Goto D
: Lbl C
: Y₂(Ø)→C
: Y₂(1)-C→D
: ((Y₂(⁻1)-C)+D)/2→A
: Y₂(1)-A-C→B
: Lbl D
: B²-4*A*C→D
: ⁻B/(2*A)→X
: A*X²+B*X+C→Y
: Disp "DISC=",D▶Frac
: Disp "ROOTS"
: If D<Ø
: Then
: Disp "2 COMPLEX"
: Disp "SOLUTIONS"
: Else
: If D=Ø
: Then
: Disp ⁻B/(2*A)▶Frac
: Else
```

```
: Disp (⁻B-√(D))/(2*A)▸Frac
: Disp (⁻B+√(D))/(2*A)▸Frac
: End
: Pause
: Disp "VERTEX (X,Y)",X▸Frac,Y▸Frac
: Stop
: Lbl 2
: Input "LIST=",L6
: min(L6)→G
: Ø→F
: While (G>1) and (F=Ø)
: 1→F
: 1→I
: While (F=1) and (I≤dim(L6))
: If fPart(L6(I)/G)≠Ø
: Ø→F
: I+1→I
: End
: If F=Ø
: G-1→G
: End
: Disp "GCF =",G
: Stop
: Lbl 3
: Prompt A
: Prompt B
: Prompt C
: abs(A*C)→D
: If fPart(D)≠Ø
: Then
: Disp "MUST BE INTEGERS"
: Goto G
: End
: Ø→T
: 1→P
: While (T=Ø) and (P≤iPart(√(D))
: D/P→Q
: If fPart(Q)=Ø
: Then
: If (C>Ø) and (P+Q=abs(B))
: 1→T
```

```
: If (C<Ø) and (Q-P=abs(B))
: 1→T
: End
: If T=Ø
: P+1→P
: End
: If T=Ø
: Then
: Disp "PRIME"
: Goto G
: End
: min(A,Q)→E
: While ((fPart(A/E)≠Ø) or (fPart(Q/E)≠Ø)) and E>1
: E-1→E
: End
: If B≤Ø
: -Q→Q
: If B*C<Ø
: -P→P
: ClrHome
: Disp " "
: Disp " "
: Output(1,1,"(")
: If E≠1
: Then
: Output(1,2,E)
: iPart(log(E))+4→F
: Else
: 3→F
: End
: Output(1,F-1,"X")
: If P*E/A≥Ø
: Then
: Output(1,F,"+")
: Else
: Output(1,F,"-")
: End
: F+1→F
: Output(1,F,abs(P*E/A))
: F+iPart(log(abs(P*E/A)))+3→F
: Output(1,F-2,")(")
```

```
: If A/E≠1
: Then
: Output(1,F,A/E)
: F+iPart(log(A/E))+2→F
: Else
: F+1→F
: End
: Output(1,F-1,"X")
: If Q/E≥Ø
: Then
: Output(1,F,"+")
: Else
: Output(1,F,"-")
: End
: F+1→F
: Output(1,F,abs(Q/E))
: F+iPart(log(abs(Q/E)))+1→F
: Output(1,F,")")
: Lbl G
: Stop
: Lbl 4
: Disp " "
: Disp "ENTER LIST OF"
: Disp "COEFF IN BRACES"
: Input "P(X)=",L₁
: Input "X   =",C
: L₁→L₂
: For(I,2,dim(L₁))
: C*L₂(I-1)+L₂(I)→L₂(I)
: End
: Pause L₂
: Stop
: Lbl 5
: Input "COEFF=",L₅
: dim(L₅)→C
: {C-1,C}→dim([E])
: For(I,1,C-1)
: If I>1
: Then
: Input "COEFF=",L₅
: End
```

```
: For(J,1,C)
: L5(J)→[E](I,J)
: End
: End
: [E]→[D]
: {C-1,C-1}→dim([D])
: det([D])→D
: C-1→dim(L6)
: For(I,1,C-1)
: [E]→[D]
: {C-1,C-1}→dim([D])
: For(J,1,C-1)
: [E](J,C)→[D](J,I)
: End
: det([D])→L6(I)
: End
: Disp "D   =",D
: Disp "DXYZ=",L6
: Pause
: If D≠Ø
: Then
: Disp "SOL =",L6/D▸Frac
: Else
: If sum(abs(L6))=Ø
: Then
: Disp "INDETERMINANT"
: Else
: Disp "NO SOLUTION"
: End
: End
: Lbl 6
: Stop
```

ANOVA

Generates a one-way analysis of variance table. The means should be entered into L1, the variances into L2, and the sample sizes into L3. Will offer to run Scheffe (unbalanced design) or Tukey (balanced design) post-hoc tests.

```
: ClrHome
```

```
: Disp "VERIFY THAT"
: Disp "x̄ IS IN L₁,"
: Disp "Sx² IN L₂, and "
: Disp "n IN L₃."
: Disp " "
: Input "1=YES, Ø=NO ?",Y
: Disp " "
: If Y≠1
: Stop
: If (dim(L₁)≠dim(L₂)) or (dim(L₂)≠dim(L₃))
: Then
: Disp "L₁, L₂, and L₃"
: Disp "MUST HAVE THE"
: Disp "SAME SIZE."
: Stop
: End
: sum(L₃*L₁)/sum(L₃)→G
: Disp "GRAND MEAN =    ",G
: Pause
: sum(L₃*(L₁-G)²)→A
: A/(dim(L₁)-1)→B
: Disp "SS(B)=",A
: Disp "DF(B)=",dim(L₁)-1
: Disp "MS(B)=",B
: Pause
: sum((L₃-1)*L₂)→C
: C/sum(L₃-1)→W
: Disp "SS(W)=",C
: Disp "DF(W)=",sum(L₃-1)
: Disp "MS(W)=",W
: Pause
: Disp "SS=",A+C
: Disp "DF=",sum(L₃)-1
: Disp "F=",B/W
: Pause
: Disp " "
: If min(L₃)=max(L₃)
: Then
: 1→D
: (√(W/L₃(1)))⁻¹→K
: Disp "RUN TUKEY ?"
```

```
: Else
: Ø→D
: Disp "RUN SCHEFFE ?"
: End
: Disp " "
: Input "1=YES Ø=NO ?",Y
: Disp " "
: If Y≠1
: Stop
: Ø→Z
: Ø→R
: For(I,1,dim(L₁)-1,1)
: For(J,I+1,dim(L₁),1)
: If Z=Ø
: ClrHome
: If D=Ø
: Then
: (L₁(I)-L₁(J))² /(W*(L₃(I)⁻¹+L₃(J)⁻¹))→T
: Else
: (L₁(I)-L₁(J))*K→T
: End
: Disp "( , )"
: Disp "TS="
: Output(2*Z+1,2,I)
: Output(2*Z+1,4,J)
: Output(2*Z+2,4,T)
: Z+1→Z
: R+1→R
: If (Z=3) and (R≠(dim(L₁) nCr 2))
: Then
: Pause
: Ø→Z
: End
: End
: End
```

BINOMIAL

Calculate binomial probabilities

```
: Input "N = ",N
```

```
: Input "P = ",P
: Input "X = ",X
: Ø→A
: Menu("CHOOSE X
  TYPE","=",A,"≥",B,"≤",C,">",D,"<",E)
: Lbl A
: (N nCr X)*P^X*(1-P)^(N-X)→A
: Goto Z
: Lbl B
: For(R,X,N,1)
: A+(N nCr R)*P^R*(1-P)^(N-R)→A
: End
: Goto Z
: Lbl C
: For(R,Ø,X,1)
: A+(N nCr R)*P^R*(1-P)^(N-R)→A
: End
: Goto Z
: Lbl D
: For(R,X+1,N,1)
: A+(N nCr R)*P^R*(1-P)^(N-R)→A
: End
: Goto Z
: Lbl E
: For(R,Ø,X-1,1)
: A+(N nCr R)*P^R*(1-P)^(N-R)→A
: End
: Lbl Z
: Disp "PROB = ",A
```

CONTING

Calculate the chi-square test statistic for the contingency table contained in matrix [A].

```
: [A]→[E]
: dim([A])→L6
: L6(1)→dim(L3)
: L6(2)→dim(L4)
: For(I,1,L6(1),1)
```

```
: Ø→X
: For(J,1,L6(2),1)
: X+[A](I,J)→X
: End
: X→L3(I)
: End
: For(J,1,L6(2),1)
: Ø→X
: For(I,1,L6(1),1)
: X+[A](I,J)→X
: End
: X→L4(J)
: End
: sum(L3)→X
: L6(1)*L6(2)→dim(L1)
: L6(1)*L6(2)→dim(L2)
: For(I,1,L6(1),1)
: For(J,1,L6(2),1)
: (I-1)*L6(2)+J→Y
: [A](I,J)→L1(Y)
: L3(I)*L4(J)/X→L2(Y)
: End
: End
: sum((L1-L2)²/L2)
```

DECISION

Takes the payoff matrix in [A] and computes the decision under the Expected Value, Maximax, Maximin, and Minimax decision criteria. Matrix [A] should have each state of nature as a row and each action as a column. The probabilities for the expected value criteria should be contain in L1.

```
: Disp "THE PAYOFF TABLE"
: Disp "SHOULD BE IN [A]"
: dim([A])→L2
: L2(1)→R
: L2(2)→C
: C→dim(L2)
: C→dim(L3)
: C→dim(L4)
```

```
: C→dim(L5)
: {R,C}→dim([E])
: For(I,1,R,1)
: [A](I,1)→X
: For(J,2,C,1)
: If [A](I,J)>X
: [A](I,J)→X
: End
: For(J,1,C,1)
: X-[A](I,J)→[E](I,J)
: End
: End
: For(J,1,C,1)
: L1(1)*[A](1,J)→L2(J)
: [A](1,J)→L3(J)
: [A](1,J)→L4(J)
: [E](1,J)→L5(J)
: For(I,2,R,1)
: L2(J)+[A](I,J)*L1(I)→L2(J)
: If [A](I,J)>L3(J)
: [A](I,J)→L3(J)
: If [A](I,J)<L4(J)
: [A](I,J)→L4(J)
: If [E](I,J)>L5(J)
: [E](I,J)→L5(J)
: End
: End
: Disp "EXP VALUE"
: Disp max(L2)
: Pause L2
: Disp "MAXIMAX"
: Disp max(L3)
: Pause L3
: Disp "MAXIMIN"
: Disp max(L4)
: Pause L4
: Disp "MINIMAX"
: Disp min(L5)
: L5
```

FINANCE

Solve simple interest, compound interest, future value, and present value problems.

```
: Lbl Ø
: Menu("FINANCE","SIMPLE INT",1,"COMPOUND
  INT",2,"FUTURE VALUE",3,"PRESENT
  VALUE",4,"QUIT",5)
: Lbl 1
: Menu("SIMPLE
  INT","INTEREST",A,"PRINCIPAL",B,"RATE",C,"TIME",D,
  "QUIT",Ø)
: Lbl 2
: Menu("COMPOUND INT","AMOUNT",G,"PRINCIPAL",H,"PER.
  RATE",I,"PERIODS",J,"EFF. RATE",K,"QUIT",Ø)
: Lbl 3
: Menu("FUTURE VALUE","FUTURE
  VALUE",N,"PAYMENT",O,"PERIODS",P,"RATE",Q,"QUIT",Ø
  )
: Lbl 4
: Menu("PRESENT VALUE","PRESENT
  VALUE",T,"PAYMENT",U,"PERIODS",V,"RATE",W,"QUIT",Ø
  )
: Lbl 5
: Stop
: Lbl A
: Input "PRINCIPL=",P
: Input "RATE    =",R
: Input "TIME    =",T
: round(P*R*T,2)→I
: Disp "INTEREST="
: Pause I
: Goto 1
: Lbl B
: Input "INTEREST=",I
: Input "RATE    =",R
: Input "TIME    =",T
: round(I/(R*T),2)→P
: Disp "PRINCIPL="
```

```
: Pause P
: Goto 1
: Lbl C
: Input "PRINCIPL=",P
: Input "INTEREST=",I
: Input "TIME    =",T
: I/(P*T)→R
: Disp "RATE    ="
: Pause R
: Goto 1
: Lbl D
: Input "PRINCIPL=",P
: Input "INTEREST=",I
: Input "RATE    =",R
: round(I/(P*R),2)→T
: Disp "TIME    ="
: Pause T
: Goto 1
: Lbl G
: Input "PRINCIPL=",P
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(P*(1+I)^N,2)→A
: Disp "AMOUNT   ="
: Pause A
: Goto 2
: Lbl H
: Input "AMOUNT  =",A
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(A/(1+I)^N,2)→P
: Disp "PRINCIPAL="
: Pause P
: Goto 2
: Lbl I
: Input "PRINCIPL=",P
: Input "AMOUNT  =",A
: Input "PERIODS =",N
: Nˣ√(A/P)-1→I
: Disp "PER RATE="
: Pause I
```

```
: Goto 2
: Lbl J
: Input "PRINCIPL=",P
: Input "AMOUNT  =",A
: Input "PER RATE=",I
: round(ln(A/P)/ln(1+I),2)→N
: Disp "PERIODS ="
: Pause N
: Goto 2
: Lbl K
: Input "NOM RATE=",R
: Input "PMTS/YR =",M
: (1+R/M)^M-1→E
: Disp "EFF RATE="
: Pause E
: Goto 2
: Lbl N
: Input "PAYMENT =",A
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(A*((1+I)^N-1)/I,2)→F
: Disp "FUT VAL ="
: Pause F
: Goto 3
: Lbl O
: Input "FUT VAL =",F
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(F*I/((1+I)^N-1),2)→A
: Disp "PAYMENT ="
: Pause A
: Goto 3
: Lbl P
: Input "PAYMENT =",A
: Input "FUT VAL =",F
: Input "PER RATE=",I
: round(ln(1+F*I/A)/ln(1+I),2)→N
: Disp "PERIODS ="
: Pause N
: Goto 3
: Lbl Q
```

```
: Input "PAYMENT =",A
: Input "FUT VAL =",F
: Input "PERIODS =",N
: solve(((1+I)^N-1)/I-F/A,I,N^x√(F/A)-1)→I
: Disp "PER RATE="
: Pause I
: Goto 3
: Lbl T
: Input "PAYMENT =",A
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(A*(1-(1+I)^⁻N)/I,2)→P
: Disp "PRES VAL="
: Pause P
: Goto 4
: Lbl U
: Input "PRES VAL=",P
: Input "PER RATE=",I
: Input "PERIODS =",N
: round(P*I/(1-(1+I)^⁻N),2)→A
: Disp "PAYMENT ="
: Pause A
: Goto 4
: Lbl V
: Input "PRES VAL=",P
: Input "PAYMENT =",A
: Input "PER RATE=",I
: round(⁻ln(1-P*I/A)/ln(1+I),2)→N
: Disp "PERIODS ="
: Pause N
: Goto 4
: Lbl W
: Input "PRES VAL=",P
: Input "PAYMENT =",A
: Input "PERIODS =",N
: solve((1-(1+I)^(⁻N))/I-P/A,I,A*N/P-1)→I
: Disp "PER RATE="
: Pause I
: Goto 4
```

GAME

Gives the solution to a two-player, zero-sum game.

```
: Ans→[E]
: dim([E])→L6
: L6(1)→dim(L1)
: L6(2)→dim(L2)
: Fill(Ø,L1)
: Fill(Ø,L2)
: Ø→V
: L6→dim([D])
: Fill(Ø,[D])
: dim(L2)→dim(L5)
: Ø→K
: For(I,1,dim(L1))
: seq([E](I,J),J,1,dim(L2),1)→L5
: For(J,1,dim(L2))
: If min(L5)=L5(J)
: 1→[D](I,J)
: End
: If I=1 or min(L5)<K
: min(L5)→K
: End
: If K≥Ø
: Then
: Ø→K
: Else
: 1-K→K
: End
: Ø→M
: For(J,1,dim(L2))
: seq([E](I,J),I,1,dim(L1),1)→L5
: For(I,1,dim(L1))
: If max(L5)=L5(I)
: Then
: [D](I,J)+1→[D](I,J)
: If [D](I,J)>1
: 1→M
: End
: End
: End
```

```
: If M=1
: Then
: For(I,1,dim(L₁))
: For(J,1,dim(L₂))
: If [D](I,J)=2
: Then
: 1→L₁(I)
: 1→L₂(J)
: [E](I,J)→V
: End
: End
: End
: L₁/sum(L₁)→L₁
: L₂/sum(L₂)→L₂
: Goto A
: End
: Disp "NON STRICTLY"
: Disp "DETERMINED."
: dim(L₁)→dim(L₃)
: dim(L₂)→dim(L₄)
: Fill(Ø,L₃)
: Fill(Ø,L₄)
: For(H,1,2)
: dim(L₂)→dim(L₅)
: For(I,1,dim(L₁)-1)
: For(M,I+1,dim(L₁))
: If L₃(I)+L₃(M)=Ø
: Then
: seq((1-L₄(J))*([E](M,J)-
  [E](I,J)),J,1,dim(L₂),1)→L₅
: If min(L₅)≥Ø
: 1→L₃(I)
: If max(L₅)≤Ø
: 1→L₃(M)
: End
: End
: End
: If H=1
: Then
: dim(L₁)→dim(L₅)
: For(J,1,dim(L₂)-1)
```

```
: For(M,J+1,dim(L₂))
: If L₄(J)+L₄(M)=Ø
: Then
: seq((1-L₃(I))*([E](I,M)-
  [E](I,J)),I,1,dim(L₁),1)→L₅
: If min(L₅)≥Ø
: 1→L₄(M)
: If max(L₅)≤Ø
: 1→L₄(J)
: End
: End
: End
: End
: End
: Disp "RECESSIVE ROWS"
: If sum(L₃)=Ø
: Then
: Disp "            NONE"
: Else
: 1→dim(L₅)
: For(I,1,dim(L₁))
: If L₃(I)=1
: Then
: I→L₅(dim(L₅))
: dim(L₅)+1→dim(L₅)
: End
: End
: dim(L₅)-1→dim(L₅)
: Disp L₅
: End
: Disp "RECESSIVE COLS"
: If sum(L₄)=Ø
: Then
: Disp "            NONE"
: Else
: 1→dim(L₅)
: For(J,1,dim(L₂))
: If L₄(J)=1
: Then
: J→L₅(dim(L₅))
: dim(L₅)+1→dim(L₅)
```

```
: End
: End
: dim(L5)-1→dim(L5)
: Disp L5
: End
: Pause
: [E]→[D]
: 2→dim(L5)
: dim(L1)-sum(L3)→L5(1)
: dim(L2)-sum(L4)→L5(2)
: L5(1)+1→L5(1)
: L5(1)+L5(2)+1→L5(2)
: L5→dim([E])
: Fill(Ø,[E])
: 1→A
: For(I,1,dim(L1))
: If L3(I)=Ø
: Then
: 1→B
: For(J,1,dim(L2))
: If L4(J)=Ø
: Then
: K+[D](I,J)→[E](A,B)
: B+1→B
: End
: End
: A+1→A
: End
: End
: For(J,1,B-1)
: -1→[E](A,J)
: End
: For(I,1,A)
: 1→[E](I,B+I-1)
: If I<A
: 1→[E](I,A+B)
: End
: If A=3 and B=3
: Then
: [E](1,1)-K→A
: [E](1,2)-K→B
```

```
: [E](2,1)-K→C
: [E](2,2)-K→D
: (A+D)-(B+C)→F
: (A*D-B*C)/F→V
: 1→G
: For(I,1,dim(L₁))
: If L₃(I)=Ø
: Then
: If G=1
: Then
: (D-C)/F→L₁(I)
: G+1→G
: Else
: (A-B)/F→L₁(I)
: End
: End
: End
: 1→G
: For(I,1,dim(L₂))
: If L₄(I)=Ø
: Then
: If G=1
: Then
: (D-B)/F→L₂(I)
: G+1→G
: Else
: (A-C)/F→L₂(I)
: End
: End
: End
: Goto A
: End
: Disp "TABLEAU"
: Pause [E]
: prgmSIMPLEX
: Pause [E]▸Frac
: [E](A,A+B)⁻¹→V
: 1→C
: For(I,1,dim(L₁))
: If L₃(I)=Ø
: Then
```

```
: V*[E](A,B+C-1)→L1(I)
: C+1→C
: End
: End
: A→dim(L5)
: A→dim(L6)
: seq([E](J,A+B),J,1,A,1)→L6
: 1→C
: For(I,1,dim(L2))
: If L4(I)=Ø
: Then
: seq([E](J,C),J,1,A,1)→L5
: If abs((sum(L5))*(sum(L5)²)-1)<E⁻1Ø
: V*sum(L5*L6)→L2(I)
: C+1→C
: End
: End
: V-K→V
: Lbl A
: Disp "ROW"
: Disp L1▶Frac
: Disp "COL"
: Disp L2▶Frac
: Disp "VALUE"
: Disp V▶Frac
```

HYPERGEO

Find probabilities from the hypergeometric distribution

```
: Disp "{N1,N2,...}"
: Input L1
: Disp "{R1,R2,...}"
: Input L2
: If dim(L1)≠dim(L2)
: Then
: Disp "DIM MUST BE SAME"
: Goto Z
: End
: 1→N
: For(I,1,dim(L1),1)
```

```
: If (L₁(I)<L₂(I)) or (L₂(I)<Ø)
: Then
: Disp "INVALID LIST"
: Goto Z
: End
: N*(L₁(I) nCr L₂(I))→N
: End
: sum(L₁) nCr sum(L₂)→D
: Disp N
: Disp "       ---------"
: Disp D
: Disp "="
: N/D→P
: Disp P▸Frac
: Lbl Z
```

LEONTIEF

Give the solution to a Leontief Input-Output Model problem. The technology matrix should be in [A] and the final demand matrix in [D].

```
: ClrHome
: Disp "VERIFY THAT THE"
: Disp "TECH MATRIX IS"
: Disp "IN [A] and THE"
: Disp "DEMAND IN [D]"
: Disp " "
: Input "1=YES Ø=NO ?",Y
: Disp " "
: If Y≠1
: Then
: Stop
: End
: dim([A])→L₆
: If L₆(1)≠L₆(2)
: Then
: Disp "[A] MUST BE A"
: Disp "SQUARE MATRIX"
: Stop
: End
: (identity(L₆(1))-[A])⁻¹*[D]▸Frac
```

MEDIAN

Find the median for classed data.

```
: Input "N     = ",N
: Disp "CF OF PREVIOUS"
: Input "CLASS = ",C
: Disp "FREQ OF MED"
: Input "CLASS = ",F
: Input "WIDTH = ",W
: Disp "LOWER BOUNDARY"
: Disp "OF MEDIAN"
: Input "CLASS = ",L
: Disp "MEDIAN= ",(N/2-C)/F*W+L
```

NUMINT

Use numerical integration to approximate the area under a curve. Uses the function in Y1 and can find the area using rectangles (left hand endpoints, right hand endpoints, or midpoints), the trapezoid method, or Simpson's method.

```
: Input "LOWER LIM= ",A
: Input "UPPER LIM= ",B
: Input "INTERVALS= ",N
: (B-A)/N→C
: Menu("TECHNIQUE","RECTANGLES",A,"TRAPEZOID",B,"SIM
  PSONS",C)
: Lbl A
: Menu("ENDPOINT","LEFT HAND",F,"RIGHT
  HAND",G,"MIDPOINT",H)
: Lbl F
: C*sum(Y₁(seq(A+K*C,K,Ø,N-1,1)))→S
: Goto Z
: Lbl G
: C*sum(Y₁(seq(A+K*C,K,1,N,1)))→S
: Goto Z
: Lbl H
: C*sum(Y₁(seq(A+Ø.5C+K*C,K,Ø,N-1,1)))→S
: Goto Z
: Lbl B
```

```
: Y₁(seq(A+K*C,K,Ø,N,1))→L₅
: C/2*(2*sum(L₅)-L₅(1)-L₅(N+1))→S
: Goto Z
: Lbl C
: If N/2≠int(N/2)
: Then
: Disp "N MUST BE EVEN"
: Disp "WHEN USING"
: Disp "SIMPSONS RULE."
: Goto Y
: End
: Y₁(seq(A+K*C,K,Ø,N,1))→L₅
: 2*sum(L₅)-L₅(1)-L₅(N+1)→S
: For(K,1,N/2)
: S+2*L₅(2*K)→S
: End
: C/3*S→S
: Lbl Z
: Disp "APPROX =",S
: Lbl Y
```

PDIST

Find the mean, variance, and standard deviation of a probability distribution. The values of the random variable *x* should be in L1 and the probabilities of those should be in L2.

```
: ClrHome
: Disp "VERIFY THAT"
: Disp "X IS IN L₁ and "
: Disp "P(X) IN L₂."
: Disp " "
: Input "1=YES Ø=NO ?",Y
: Disp " "
: If Y=1
: Then
: If dim(L₁)=dim(L₂)
: Then
: If min(L₂)<Ø or max(L₂)>1
: Then
: Disp "ERROR: VERIFY"
```

```
: Disp " Ø ≤ P(X) ≤ 1"
: Disp "BY EDITING L₂"
: Goto A
: End
: sum((L₁*L₂)→M
: sum((L₁²*L₂)→Z
: Z-M²→V
: Disp "SUM P(X)  = ",sum(L₂)
: Disp "SUM XP(X) = ",M
: Disp "SUM X²P(X)= ",Z
: Pause
: If V<Ø
: Then
: Disp "ERROR: VERIFY"
: Disp " SUM P(X)=1"
: Disp "BY EDITING L₂"
: Else
: √(V)→S
: Disp "MEAN    = ",M
: Disp "VAR     = ",V
: Disp "STDEV   = ",S
: End
: Else
: Disp "THE LISTS MUST"
: Disp "BE THE SAME SIZE"
: End
: End
: Lbl A
```

PIE

Create a pie chart from the frequencies in L1

```
: Ø→S
: If dim(L₁)>Ø
: sum(L₁)→S
: If S=Ø
: Then
: Disp "PUT THE FREQS"
: Disp "IN L₁"
: Stop
```

```
: End
: ClrHome
: Disp "LABEL?"
: Disp "  1=YES"
: Disp "  Ø=NO"
: Input "(Ø/1)? ",L
: Radian
: AxesOff
: ZStandard
: ZSquare
: Circle(Ø,Ø,1Ø)
: Ø→T
: 2*π/S→S
: Line(Ø,Ø,1Ø,Ø)
: For(I,1,dim(L₁),1
: T+L₁(I)→T
: Line(Ø,Ø,1Ø*cos(T*S),1Ø*sin(T*S)
: If L=1
: Then
: T-Ø.5*L₁(I)→U
: Text(int(29-
  6*YFact*sin(U*S)),int(47+6*XFact*cos(U*S)),L₁(I))
: End
: End
```

PIVOT

Pivot on a row and column in a matrix. This is an interface to the PIVOT1 program that actually does the work. The matrix to pivot on should be ANS before running the program.

```
: Ans→[E]
: Input "PIVOT ROW =",R
: Input "PIVOT COL =",C
: dim([E])→L₆
: L₆(1)→M
: L₆(2)→N
:
: prgmPIVOT1
:
: [E]▸Frac
```

PIVOT1

Manipulate a matrix by making the pivot element a 1 and clearing out the pivot column. Performs its work on matrix [E] and pivots on row R and column C. This used by the PIVOT and SIMPLEX programs and should never be called directly.

```
: If (R>M) or (C>N)
: Then
: Disp "PIVOT ELEMENT"," OUTSIDE MATRIX"
: Pause
: Goto F
: End
: If [E](R,C)=Ø
: Then
: Disp "CANNOT PIVOT ON"," A ZERO"
: Pause
: Goto F
: End
: *row(([E](R,C))⁻¹,[E],R)→[E]
: For(I,1,M)
: If (I≠R) and ([E](I,C)≠Ø)
: *row+(⁻[E](I,C),[E],R,I)→[E]
: End
: For(I,1,M)
: For(J,1,N)
: If abs([E](I,J))≤E⁻12
: Ø→[E](I,J)
: End
: End
: Lbl F
```

QQPLOT

This program generates a normal quantile-quantile plot (also known as a normal probability plot) of the data in L1.

```
: L₁→L₆
: dim(L₆)→N
: N→dim(L₅)
```

```
: SortA(L6)
: For(I,1,N,1)
: I/(N+1)→P
: P*(1-P)→C
: If C<E-8
: Then
: 5→Z
: Else
: -ln((4*C))→Y
: √((Y*(2.Ø611786-5.7262Ø4/(Y+11.64Ø595))))→Z
: End
: If P<Ø.5
: -Z→Z
: Z→L5(I)
: End
: Plot1(Scatter,L5,L6,▫)
: ZoomStat
```

SERIES

Work with arithmetic and geometric sequences finding the general term or the sum of the series.

```
: Lbl Ø
: Menu("SERIES","ARITHMETIC",1,"GEOMETRIC",2,"QUIT",
  3)
: Lbl 1
: Menu("ARITHMETIC","GENERAL TERM",A,"SUM
  A(1),A(N)",B,"SUM A(1),N,D",C,"QUIT",Ø)
: Lbl 2
: Menu("GEOMETRIC","GENERAL TERM",E,"FINITE
  SUM",F,"INFINITE SUM",G,"QUIT",Ø)
: Lbl 3
: Stop
: Lbl A
: Input "A(1) = ",A
: Input "N    = ",N
: Input "D    = ",D
: Disp "A(N) ="
: Pause A+(N-1)*D
: Goto 1
```

```
: Lbl B
: Input "A(1) = ",A
: Input "A(N) = ",B
: Input "N    = ",N
: Disp "S ="
: Pause N*(A+B)/2
: Goto 1
: Lbl C
: Input "A(1) = ",A
: Input "N    = ",N
: Input "D    = ",D
: Disp "S    ="
: Pause N*(2*A+(N-1)*D)/2
: Goto 1
: Lbl E
: Input "A(1) = ",A
: Input "N    = ",N
: Input "R    = ",R
: Disp "A(N) ="
: Pause A*R^(N-1)
: Goto 2
: Lbl F
: Input "A(1) = ",A
: Input "N    = ",N
: Input "R    = ",R
: Disp "S    ="
: Pause A*(1-R^N)/(1-R)
: Goto 2
: Lbl G
: Input "A(1) = ",A
: Input "R    = ",R
: If abs(R)≥1
: Then
: Pause "SERIES DIVERGES"
: Goto 2
: Else
: Disp "S    ="
: Pause A/(1-R)
: Goto 2
```

SIMPLEX

This program performs the SIMPLEX procedure to solve a standard maximization linear programming problem. The tableau should be ANS before running the program.

```
: Ans→[E]
: dim([E])→L6
: L6(1)→M
: L6(2)→N
: min(seq([E](M,I),I,1,N-2,1))→Z
: If Z≥Ø
: Goto D
: Lbl A
: -1→P
: Ø→R
: For(I,1,N-2)
: If [E](M,I)=Z
: Then
: -1→T
: For(J,1,M-1)
: If ([E](J,I)>Ø) and ([E](J,N)≥Ø)
: Then
: [E](J,N)/[E](J,I)→Q
: If (T=-1) or (Q<T)
: Then
: Q→T
: J→L
: End
: End
: End
: If T>P
: Then
: I→C
: L→R
: T→P
: End
: End
: End
: If R=Ø
: Then
: Disp "UNBOUNDED"
```

```
: Pause
: Else
: Disp "PIVOT ROW",R
: Disp "PIVOT COL",C
: Pause
:
: prgmPIVOT1
:
: min(seq([E](M,I),I,1,N-2,1))→Z
: If Z≥Ø
: Goto D
: Pause [E]▸Frac
: Goto A
: End
: Lbl D
: Disp "FINAL TABLEAU"
: [E]▸Frac
```

SIMULAT

Conduct a simulation of flipping one or more coins or rolling one or more dice. Will also generate numbers using a uniform distribution and then make a histogram of the results.

```
: Menu("SIMULATIONS","COINS",C,"DICE",D,"UNIFORM",U,
  "HISTOGRAM",H)
: Lbl C
: Input "NUM COINS ",R
: Input "NUM FLIPS ",N
: seq(I,I,Ø,R,1)→L₁
: L₁→L₂
: Fill(Ø,L₂)
: For(I,1,N,1)
: Ø→X
: For(J,1,R,1)
: X+int(rand*2)→X
: End
: L₂(X+1)+1→L₂(X+1)
: End
: Goto Z
: Lbl D
```

```
: Input "NUM DICE  ",R
: Input "NUM ROLLS ",N
: seq(I,I,R,6*R,1)→L₁
: L₁→L₂
: Fill(Ø,L₂)
: For(I,1,N,1)
: Ø→X
: For(J,1,R,1)
: X+int(rand*6)→X
: End
: L₂(X+1)+1→L₂(X+1)
: End
: Goto Z
: Lbl U
: Input "MINIMUM ",A
: Input "MAXIMUM ",B
: Input "SIZE    ",N
: seq(I,I,A,B,1)→L₁
: L₁→L₂
: Fill(Ø,L₂)
: For(I,1,N,1)
: int(rand*(B-A+1))→X
: L₂(X+1)+1→L₂(X+1)
: End
: Goto Z
: Lbl H
: Menu("HISTOGRAM","ON",I,"OFF",J)
: Lbl I
: FnOff
: PlotsOff
: PlotsOn 3
: Plot3(Histogram,L₁,L₂)
: min(L₁)→Xmin
: max(L₁)+1→Xmax
: 1→Xscl
: max(L₂)-min(L₂)→R
: ⁻R/12→Ymin
: 13*R/12→Ymax
: If R>2Ø
: Then
: 5→Yscl
```

```
: Else
: 1→Yscl
: End
: DispGraph
: Stop
: Lbl J
: PlotsOff
: Stop
: Lbl Z
: L₂
```

STDNORM

Find the area to between 0 and z or return the z-value when the area in the tail is known.

```
: Menu("NORMAL","KNOW ALPHA",A,"KNOW Z",Z,"QUIT",Q)
: Lbl A
: Input "ALPHA = ",P
: P*(1-P)→C
: If C<E⁻8
: Then
: 5→Z
: Else
: ⁻ln(4*C)→Y
: √(Y*(2.Ø611786-5.7262Ø4/(Y+11.64Ø595)))→Z
: End
: Disp Z
: Goto Q
: Lbl Z
: Input "Z = ",Z
: fnInt(e^(⁻.5*X²),X,Ø,Z)/√(2*π)→P
: Disp P
: Lbl Q
```

TWOWAY

Performs a balanced two-way analysis of variance test on the data in [A]. Each column represents a factor and that replications are spread across rows.

```
: Disp "VERIFY DATA IS"
```

```
: Disp "IN [A] and THAT"
: Disp "EACH TREATMENT"
: Disp "GROUP IS 1 COL"
: Disp "and MANY ROWS"
: Disp " "
: Disp "2=TRANSPOSE [A]"
: Input "1=YES Ø=NO ?",Y
: Disp " "
: If Y=Ø
: Stop
: If Y=2
: [A]ᵀ→[A]
: dim([A])→L₆
: L₆(2)→B
: Disp "NUMBER OF"
: Disp "REPLICATIONS"
: Input N
: L₆(1)/N→A
: If A≠int(A)
: Then
: Disp "INVALID DIM."
: Disp "NUMBER OF"
: Disp "REPLICATIONS"
: Disp "MUST BE A"
: Disp "FACTOR OF",L₆(1)
: Stop
: End
: Ø→S
: A→dim(L₁)
: B→dim(L₂)
: A*B→dim(L₅)
: A*B→dim(L₆)
: Fill(Ø,L₁)
: Fill(Ø,L₂)
: Fill(Ø,L₅)
: Fill(Ø,L₆)
: For(I,1,A,1)
: For(J,1,B,1)
: For(K,1,N,1)
: N*(I-1)+K→R
: [A](R,J)→Z
```

```
: L₁(I)+Z→L₁(I)
: L₂(J)+Z→L₂(J)
: L₅(B*(I-1)+J)+Z→L₅(B*(I-1)+J)
: L₆(B*(I-1)+J)+Z²→L₆(B*(I-1)+J)
: End
: End
: End
: (sum(L₁))/(A*B*N)→G
: L₁/(N*B)→L₁
: L₂/(N*A)→L₂
: L₅/N→L₅
: N*B*sum((L₁-G)²)→R
: N*A*sum((L₂-G)²)→C
: sum(L₆-L₅²*N)→W
: Ø→Z
: For(I,1,A,1)
: For(J,1,B,1)
: (L₅((I-1)*B+J)-L₁(I)-L₂(J)+G)²→Y
: Z+Y→Z
: End
: End
: N*Z→I
: W/(A*B*(N-1))→M
: Disp "SS(R) = ",R
: Disp "DF(R) = ",A-1
: Disp "MS(R) = ",R/(A-1)
: Disp " "
: Pause
: Disp "SS(C) = ",C
: Disp "DF(C) = ",B-1
: Disp "MS(C) = ",C/(B-1)
: Disp " "
: Pause
: Disp "SS(I) = ",I
: Disp "DF(I) = ",(A-1)*(B-1)
: Disp "MS(I) = ",I/((A-1)*(B-1))
: Disp " "
: Pause
: Disp "SS(W) = ",W
: Disp "DF(W) = ",A*B*(N-1)
: Disp "MS(W) = ",W/(A*B*(N-1))
```

```
: Disp " "
: Pause
: Disp "SS(T) = ",R+C+I+W
: Disp "DF(T) = ",A*B*N-1
: Disp " "
: Disp " "
: Disp " "
: Pause
: Disp "F(R)  = ",R/(A-1)/M
: Disp "F(C)  = ",C/(B-1)/M
: Disp "F(I)  = ",I/((A-1)*(B-1))/M
```