Functions	Syntax	Uses
1. **Asymptotes (Asymp)**	asymp(*function*, *variable*)	Determines the vertical and horizontal asymptotes of a function.
2. **Composite Function Check (ccheck)**	ccheck(Outer function, inner function)	Determines if a composite exists, and if not, determines the maximal domain for which a composite exists.
3. **Discriminant (Discrim)**	discrim(*function*, *variable*, 15)	Calculates the discriminant of an inputted quadratic expression.
4. **Domain and Range (Domrang)**	domrang(*function*, *variable*)	Determines the domain and range of a function. **Note:** Due to CAS approximation bounds may not be entirely accurate.
5. **Intercepts (Intercepts)**	intercepts(*function*,*variable*)	Finding the x and y intercepts of a function.
6. **Intersects (Intersects)**	intersects(*function1*,*function2*,*variable*)	Determines the points of intersection of two functions across their maximal domains.
7. **Intersects with domain (intersectsd)**	intersectsd(*function1*, *function2*, *variable*, *lower*, *upper*)	Determines the points of intersection between two functions in a restricted domain.
8. **Inverse Function (inverse)**	inverse(*function, variable, x in domain of f*)	Determines the inverse of a given function.
9. **Inverse Intersections (invents)**	inverse(function, number of intersections with inverse)	Determines the values of a parameter, k, required for a function and its inverse to have a specified number of intersections. Works with: logs, exponentials, sqrt, parabolas, cubics, hyperbolas.
10. **Angle between two lines (lineang)**	lineang(*Line 1, Line 2, Variable*)	Determines the angle between two lines, and assumes your CAS is in **radians** mode not degrees mode.
11. **Unique, None, Infinite Solution (Linesolve)**	linesolve(*Equation1, Equation2*)	Determines when two equations will have an unique, none or infinitely many solutions.
12. **Property Check (pcheck)**	pcheck(*function, variable, LHS, RHS*)	Determines which function satisfies a specific property. **Note:** You **must** define the function outside of the program.
13. **Point Information (pointinfo)**	pointinfo(x_1, y_1, x_2, y_2)	Determines the gradient, perpendicular gradient, line, x and y intercepts of a line, midpoint, distance.
14. **Polynomial Fit (polyfit)**	polyfit($\{x_1, y_1, ...\}$)	Determines a polynomial equation which passes through all the inputted pairs of points.
15. **Transformations (transform)**	transform(*function*, {*transformations*})	Determines the transformed function after applying certain transformations. **Note:** The transformations do not use real math, for example, $2y$ corresponds to a dilation by a factor of 2 from the y axis in the program.

Calculus	Syntax	Uses
1. **Average Rate of Change (avgroc)**	avgroc(*function*, *variable*, *lower*, *upper*)	Determines the average rate of change of a function.
2. **Average Value (avgval)**	avgval(*function*, *variable*, *lower*, *upper*)	Calculates the average value of a function.
3. **Bound Area (boundarea)**	boundarea(*function1, function2, variable*)	Determines the area bound by two graphs (if any) across their maximal domains.
4. **Bound Area with domain (boundaread)**	boundaread(*function1, function2, variable, lower, upper*)	Determines the bound area between two functions in a restricted domain.
5. **Integral Guess (intguess)**	**One** integral given, find **transformed** integral. intguess({*lower1,upper1, value1*}, {*transformations*}, {*lower2, upper2*}) **Two** integrals given, find another integral of **untransformed** function. intguess({*lower1, upper1, value1*}, {*lower2, upper2, value2*}, {*lower3, upper3*}) **Case 3: Two** integrals given, then find another integral of **transformed** function. intguess({*lower1, upper1, value1, lower2, upper2, value2*}, {*transformations*}, {*lower3, upper3*})	Determines the answer for the integration multiple choice questions.
6. **Newton's Method (newtons)**	newtons(*function*, *variable*, x_0, *iterations*)	Estimates the root of a function using newton's method.
7. **Number of Roots (nroot)**	nroot(*polynomial, variable, parameter, no. roots*)	Determines the value(s) of the parameter for which a **polynomial** will have n roots.
8. **Number of Stationary Points (nstp)**	nstp(*polynomial, variable, parameter, no. stp*)	Determines the value(s) of the parameter for which a **polynomial** will have n stationary points.
9. **Number of Points of Inflection (nstp)**	npoi(*polynomial, variable, parameter, no. stp*)	Determines the value(s) of the parameter for which a **polynomial** will have n points of inflection.
10. **Points of Inflection (pois)**	pois(*function*, *variable*)	Determines the points of inflection of a function.
11. **Sign Table (signtab)**	signtab(*function, variable*)	Uses a sign table to determine the nature of the stationary points of a function.
12. **Stationary Points (stps)**	stps(*function, variable*)	Determines the stationary points of a function.
13. **Tangent Solve (tangsolve)**	tangsolve(*function, variable*, x_0, y_0)	Determines the equation of the tangents of the function which pass through the specified point.
14. **Trapezoid Approximation (trapapprox)**	trapapprox(*function, variable, lower, upper, number of trapezia*)	Approximates an integral using the trapezoidal rule.

Continuous Probability	Syntax	Uses
1. **Continuous Conditional Probability (ccondpr)**	**Probability Density Function:** ccondpr(*Probability Density Function*, *Lower Bound, Upper Bound*) **Normal Distribution:** ccondpr(" ", *Mean*, *Standard Deviation*, Condition 1, Condition 2)	Determines conditional probability for a continuous distribution.
2. **Confidence Interval (confint)**	confint(*Sample Size*, $\hat{P}$, *. confidence*)	Determines a confidence interval as well as the z-score, margin of error and standard deviation.
3. **Confidence Interval Solve (confintsolve)**	confintsolve(*Lower Bound, Upper Bound, Sample Size or Sample Standard Deviation or . Confidence)*	Determines the sample size, standard deviation or percentage confidence depending on the provided data. **Note:** This program assumes no confidence levels less than 50% will even be used.
4. **Continuous Distribution Information (continfo)**	continfo(*function*, *variable*, *lower*, *upper*)	Determines the expected value, mean, variance, standard deviation of a continuous probability distribution.
5. **Inverse Normal (invnormvals)**	invnormvals(*mean*, *standard deviation*, *probability*)	Determines the left, right and centre possibilities for probability of a distribution.
6. **Normal Solve (normsolve)**	**Case 1:** Both lower and upper given. normsolve(*Lower, Probability of Lower, Upper, Probability of Upper*) **Case 2:** Lower and μ given. normsolve(*Lower, Probability of Lower*, μ, " ") **Case 3:** Lower and σ given. normsolve(*Lower, Probability of Lower*, " ",σ) **Case 4:** Upper and μ given. normsolve(μ, " ", *Upper, Probability of Upper*) **Case 5:** Upper and σ given. normsolve(" ", σ, *Upper, Probability of Upper*)	Determines the mean and standard deviation for lower and upper type questions. **Note:** If required, convert into this format using complement.

Discrete Probability	Syntax	Uses
1. **Binomial Solve**	binomsolve(*outcome*, *probability of success*, *threshold value*)	Determines the number of trials required to achieve a certain probability.
2. **Discrete Conditional Probability**	**Binomial:** dcondpr(*number of trials, probability of success, condition 1, condition 2*) **Probability Table:** dcondpr({*List containing outcomes*}, {*List containing probabilities*}, condition 1, condition 2*) **Probability Mass Function:** dcondpr({*List containing outcomes*}, *PMF, condition 1, condition 2*)	Determines conditional probability for a discrete distribution.
3. **Binomial Distribution Information**	binominfo(*Sample Size*, *Probability of Success*)	Determines the expected value, variance, standard deviation, sample expected value, and sample standard deviation for a binomial distribution.
4. **Hypergeometric Cumulative Probability Function**	hypergeocdf(*sample size*, *population size*, *number of successful items*, *lower bound*, *upper bound*)	Determines the probability of selecting items without replacement, but over an interval of outcomes.
5. **Hypergeometric Probability Density Function**	hypergeopdf(*sample size*, *population size*, *number of successful items*, *outcome*)	Determines the probability of selecting items without replacement, but for specific outcomes.
6. **Inverse Binomial**	invbinomial(*number of trials*, *probability of success, known probability value*)	Determines the outcome required to achieve the probability. **Note:** Will not work if probability is too accurate. In this case just decrease accuracy by decreasing number of decimal places.
7. **Probability Table**	prtable({*outcomes*}, {*probabilities*}))	Determines the mean, variance, standard deviation of a probability table.
8. **Sample Distribution Binomial**	samplebinom(*Sample Size*, *Probability of Success*)	Determines the distribution for the sample proportion of a binomially distributed sample.
9. **Sample Binomial Probability**	samplebinompr(*Sample Size*, *Probability of Success*, *Lower*, *Upper*)	Determines the probability for the sample proportion for a binomially distributed sample.
10. **Sample Distribution Hypergeometric**	samplehypergeo(*Sample Size*, *Population Size*, *Number Successful*)	Determines the distribution for the sample proportion of a hypergeometrically distributed sample.
11. **Sample Hypergeometric probability**	samplehyppr(*Sample Size*, *Population Size*, *Number Successful*, *Lower*, *Upper*)	Determines the probability for the sample proportion for a hypergeometrically distributed sample.

Miscellaneous	Syntax	Uses
1. Column Augment	ca(Ans, {variables})	If there is a long list of x and y values, then this will convert it into an easy to read matrix form.
2. Domain Solve	dsolve(*Equation, Variable, Lower Bound, Upper Bound*)	Determines the solutions to an equation in a restricted domain.
3. Graph Information	**Maximal Domain:** graphinfo(*function*, *variable*, " ", *random*) **Restricted Domain:** graphinfo(*function*, *variable*, *lower*, *upper*)	Determines axes intercepts, stationary points, points of inflection, and endpoints of a function.
4. Trigonometric Solve	trigsolve(*Equation* or *inequality*, *variable*, *lower*, *upper*)	Determines the **exact** solutions to trigonometric equations and inequalities which the CAS cannot solve properly. For example, $\sin(x) = \cos(2x)$