- (5%) Coding Budget: You are allowed at most one `DATA` steps and three `PROC` steps to complete this assignment.

Specifications:

- (0%) Before your team starts, take a look back at your first Mini-Project of the semester. The first question asked you what you recalled about sampling distributions, what happens to the sampling distribution of the mean as the sample size increases, and what role the parent distribution plays in the sampling distribution. Take some time to discuss your answers with your current team to see what you all remember about these concepts.
- (0%) The point of the first and third mini-projects are to explore sampling distributions in two stages. Stage one is the static images your first team made for the normal and Cauchy distributions. (Recall you produced six specific graphs at the end of Mini-Project 1 - if you don't remember them, go back and look now!) Stage 2 occurs now - the purpose of Mini-Project 3 is to write a macro that produces an animation that can be used to explore the questions above about sampling distributions. (This is a statistics course, after all!) Your team's macro should: generate the data under user-specified conditions, compute all necessary values, and create the final animation. If user-provided values are invalid, your macro should also produce useful messages to help the user debug the issue. The items below detail what the macro should do, what kind of messages should be generated, and what the final animation should look like.
- (5%) Setting up paths: Programmatically change your working directory as needed to set up any `FILENAME` and `LIBNAME` statements. Do not use absolute paths anywhere in your code other than in these X commands at the top of your program!
- (5%) The only output from each execution of your program should be a single GIF. No other results should appear in any ODS destination.
- (65%) Because investigating sampling distributions can be done under a lot of conditions, and because who might use this macro have computers with different capabilities, your macro needs to allow the user to control the following items. There should be no %LET statements in your code. (To think about why that would be a bad idea, compare Questions 3, 11, 15, and 19 from the second midterm!) For each of the items below I have indicated whether your macro should consider the parameter required (user must pass a value for your macro to work) or optional (user should be able to run your macro without specifying a value). For all optional parameters, I have indicated what to use for the default value. If a parameter should only have a limited set of options, I have indicated what the acceptable values are. (I have grouped some together rather than everything getting its own line - so read carefully.) Name your macro CLTSIM
  - Required: Name of distribution. Allowable values: normal, Cauchy.
  - Optional: Seed. Allowable values: any valid seed. Default: 0.
  - Optional: Number of Samples. Default: 10.
  - Required: Minimum sample size.
  - Required: Maximum sample size.
  - Required: Filename.
  - Optional: Sample size increment. Default: 1
  - Optional: Sample statistic. Default: Mean. Allowable values: Any single summary statistic that MEANS or UNIVARIATE can calculate.
  - Optional: Graph type. Default: EmpDist (Histogram + Normal Density + Kernel Density), Box (boxplot). [As a BONUS: consider using EmpDist and boxplot simultaneously.]
  - Optional: Kernel Smoothing. Default: SAS default value for normal and 10 for Cauchy.
  - Optional: Animation format. Default: GIF. Allowable values: GIF, SVG
  - Optional: Physical Size. Default: 8in wide x 5.33in tall.
  - Optional: Animation Duration. Default:0.5 seconds/image.
  - Optional: Animation Looping. Default: No.
  - Optional: Inset. Allowable values: Yes, No. Default: Yes.
  - Optional: Density Line Thickness (for both lines). Default: 1.
  - Optional: Density Colors. Default: 9999FF for normal and CC6600 for kernel.

  - Optional: Density Line Style. Default: Solid for normal and dashed for kernel.
  - Optional: Footnote text. Default: See provided animation.
  - Optional: Text sizes. Default: 12pt for inset title and axis labels, 10pt for inset and axis values
  - Optional: ANTIALIASMAX= value. Default: 50000.
  - Optional: X-axis Range. Default: Mean ± 3*s. Options: User can specify how many standard deviations to graph on each side of the mean.

- (10%) To make matching other elements of the graph easier, make sure to do the following:
  - Decide early in your process which graphing procedure should be used.
  - Use the NOAUTOLEGEND option.
  - Use the BESTD6.2 format for the statistics in the INSET.
  - The y-axis should always be 0 to 100 by 20.
  - No matter what the user decides to use as the x-axis range, tick marks should appear in whole standard deviation units.
  - The footer is left justified and 8 point font.
  - Use the OPAQUE option on the inset.
  - The grid lines use the default styling.

- (10%) Macros of this magnitude need substantial documentation to make them easily accessible to users. For this macro, do the following:
  - In your header, include the names of every macro variable you use, a description of what it is used for, and what (if any) default value is in place.
  - If your users passes incorrect values, you'll need to print out a statement informing them: (a) the name of the parameter with an invalid value, (b) the provided value, and (c) what the valid options are.
  - Don't force your users to read your mind. For example, if they specify normal instead of Normal - your macro should account for that. Similarly, Y or Yes or y or yes should all be the same if the user is passing yes/no information.
  - Warn your user about incompatible settings like using a small sample size of 10 and large sample size of 5.
  - Warn your user if a simulation may take too long. For our case, print out a message any time the number of samples is over 100 or the simulation includes sample sizes over 100.
  - Use the prefix QC_WARNING(CLTSIM): at the beginining of all your messages to distinguish them from other notes, warnings, or errors.
  - Don't worry about invalid values for SAS syntax - SAS will issue its own messages about those!

- (0%) You should make sure your macro works under various scenarios. I encourage you to come up with some of your own - making sure to test your different error messages and combinations of other parameters until you're sure it functions as intended. We'll be checking the macro under a variety of scenarios.