Advanced Class Specs - Group 5 Hamilton (CSE 498)

- **Final Application Overview**
    - C++ tabular data loading, querying, editing and saving library (similar to Pandas' DataFrame class)
    - Special Feature: Dynamic columns using column based calculations (similar to Excel's calculated columns)
    - Class Development Focus: DataGrid and ExpressionParser
- **DataGrid**
    - Class Description
        - The updated DataGrid will allow:
            - Column-based formulas: Auto-apply expressions to entire columns
            - Filtering & selection: Query rows dynamically (SelectRows(where col1 > 50))
            - Sorting & grouping: Sort rows and group by a column for aggregations
    - Similar Classes:
        - std::vector
            - The class is a vector of vectors at its core
        - Similar to Pandas' DataFrame
    - Key Functions
        - DataGrid MergeGrids(DataGrid data_grid1, DataGrid data_grid2, bool is_column_merge)
            - Merges two DataGrid objects either column or row wise. Returns a new DataGrid containing the merged grids.
        - map<Datum, DataGrid> GroupBy(int column_index)
            - Returns a map of datagrids
            - Each key is the grouped by attribute, one of the values in the column, and the value is a new datagrid with rows that had a given grouped attribute
        - void sort(bool is_ascending_order)
            - Sorts the entire datagrid by column, prioritizing left columns over right columns
        - void sortColumn(int column_index, bool is_ascending_order)

    - Sorts a column in datagrid in ascending or descending order.
- DataGrid slice(int column_index1, int column_index2, int row_index1, int row_index1)
    - Get a slice of the datagrid buy indices
- int search(int column_index, Datum value)
    - Search for a Datum value within a column. Returns the index of the position or -1 if not found.
- CustomStruct Describe(Datagrid data_grid)
    - Mathematical summary of a datagrid outputted to the terminal and returned as a struct
- Mathematical functions:
    - double Mean(int column_index)
        - Calculates the mean for a column
    - double Median(int column_index)
        - Calculates the median for a column
    - double Mode(int column_index)
        - Calculates the mode for a column
    - double StandardDeviation(int column_index)
        - Calculates the standard deviation for a column
    - double Min(int column_index)
        - Determines min for a column
    - double Max(int column_index)
        - Determines max for a column
- Comparison Functions:
    - vector<Datum> LessThan(int column_index, Datum value)
        - Gets a sub-column of a column with values less than the desired value
    - vector<Datum> LessThanOrEqual(int column_index, Datum value)
        - Gets a sub-column of a column with values less than or equal to the desired value

                - vector<Datum> GreaterThan(int column_index, Datum value)
                    - Gets a sub-column of a column with values greater than the desired value
                - vector<Datum> GreaterThanOrEqual(int column_index, Datum value)
                    - Gets a sub-column of a column with values greater than or equal to the desired value
                - vector<Datum> Equal(int column_index, Datum value)
                    - Gets a sub-column of a column with values equal to the desired value
                - vector<Datum> NotEqual(int column_index, Datum value)
                    - Gets a sub-column of a column with values not equal to the desired value
    - Error Conditions:
        - *Programming Error*: Indexing out of bounds.
            - Throw error
        - *User Error*: Sorting doubles and string datums together.
            - May create unwanted results
    - Challenges
        - Efficiently handling merge and column operations. Especially for large DataGrids
    - Additional TODO
        - Add Column Names to DataGrid
- **ExpressionParser**
    - Class Description
        - The updated ExpressionParser will:
            - Function support: Implement sin(), cos(), log(), sqrt()
            - Custom functions: Allow users to define their own functions within expressions
    - Similar Classes
        - Nothing too similar
    - Key Functions

        - Additional Mathematical Equations:
            - auto MakeSinFun(const string name1)
                - Calculates sin of the value (sin(x))
            - auto MakeCosFun(const string name1)
                - Calculates cos of the value (cos(x))
            - auto MakeSquareRootFun(const string name)
                - Calculates square root of value (sqrt(x))
            - auto MakeExponentFun(const string name1, const string name2)
                - Calculates exponent (x^y)
        - Custom Equations - Professor recommended Pratt Parsing (I think)
            - double Evaluate(map<string, double> number_map, string equation)
                - Will use Pratt Parser method
                    - Some resources I found while researching - Note: I've only skimmed these. Definitely not an expert:
                        - https://www.youtube.com/watch?v=2l1Si4gSb9A&ab_channel=ColinJames
                        - https://journal.stuffwithstuff.com/2011/03/19/pratt-parsers-expression-parsing-made-easy/
                        - https://matklad.github.io/2020/04/13/simple-but-powerful-pratt-parsing.html
                - Will probably require many helper functions to enhance readability and simplify the code
                - If you have any advice for the Pratt Parser Professor Ofria, it would be greatly appreciated.
        - std::vector<Datum> EvaluateNewColumn(DataGrid data_grid, string equation)
            - Creates a new vector of values based on every row in data_grid and the equation.
    - Error Conditions:

        - *Programming Error*: Indexing out of bounds.
            - Throw error
        - *Programming Error*: Division by 0.
            - Throw error
    - Challenges
        - The Pratt Parser seems like it will be difficult for the order of operations for a string. Any information on this would be greatly appreciated.
    - Additionally TODO
        - Allow for different containers
            - Example: Allow lists
        - Allow for different types of containers
            - Example: List of doubles or a list of ints.