CS 2337 – PROJECT 3 – Recreating the Speed Force

Pseudocode Due: 10/25 by 11:59 PM (no late submission)

Core Implementation Due: 11/1 by 11:59 PM (no late submission)

Final Submission Due: 11/7 by 11:59 PM (submit up to 24 hours late)

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project source code will be submitted in Zybooks.
  - Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.3.0 or higher with the following flags enabled
  - -Wall
  - -Wextra
  - -Wuninitialized
  - -pedantic-errors
  - -Wconversion
- **Type your name and netID in the comments at the top of all files submitted. (-5 points)**

Objective: Use object-oriented programming and generics to implement and utilize a binary search tree.

Problem: Harrison Wells is working to create an artificial speed force for Barry Allen (The Flash) so that he can recharge. The calculations for this task involve polynomial integration. A program is needed to develop and evaluate anti-derivatives. Since it is your first day at S.T.A.R. Labs, this task has been handed to you to complete.

Pseudocode: Your pseudocode should describe the following items

- Identify the functions you plan to create for each class (including main)
  - You do not have to include pseudocode for basic items like constructors, accessors, mutators
  - Remember that the binary search tree and node classes are generic
  - Functions in these classes must be general so that they can be used in any program
- For each function, identify the following
  - Determine the parameters
  - Determine the return type
  - Detail the step-by-step logic that the function will perform
- A list of at least 10 test cases you will check during testing

Zybooks Information:

- You will have multiple source files
  - `Main.cpp`
  - `BinTree.h`
  - `Node.h`
  - `Term.h`
  - `Term.cpp`
- Core implementation has unlimited submissions
- Final submission is limited to 15 submissions

- Whitespace will not be checked

Core Implementation

- Create anti-derivatives from indefinite integrals
- Display anti-derivatives in order from highest to lowest exponent
- No evaluation of definite integrals
- Recursion not required
- Generics not required

Classes

- Use good programming practice for classes – proper variable access, mutators and accessors, proper constructors, etc.
- Remember that classes exist to be used by other people. Just because you don't use it in the program doesn't mean it shouldn't be coded and available for others to use in theirs.
- As with previous projects, you are open to design the classes as you see fit with the minimum requirements listed below
- All classes must be of your own design and implementation.
- Do not use the pre-defined Binary Search Tree class (-20 points if used)

Class Requirements

- Binary Tree class
  - Root pointer
  - Will contain functions for basic functionality of a binary tree
    - Insert (single Generic parameter)
    - Search (single Generic parameter)
    - Remove (single Generic parameter)
  - May contain other functions as long as the functions are specific to a binary search tree
  - Any function traversing the tree (partially or in full) must be recursive (-10 per function if not)
    - This includes functions that add, delete, or search the tree
- Node class
  - Generic class (-10 points if not)
  - Left and right pointers
  - Generic reference variable to hold data
- Payload class
  - Coefficient
  - Exponent

Details:

- Start the program by prompting the user for the input filename
- Read basic polynomial integrals from a file and evaluate them.
- Each term of the polynomial will be stored in a node in the tree
- For each polynomial, create the anti-derivative
- If the integral is definite, evaluate the anti-derivative with the given boundaries
- The anti-derivative is to be written in standard form (highest to lowest exponent)

Input:

- All expressions will be read from a file.
- There is no limit to the number of expressions that can be in the file.
- Each expression will be on a separate line.
- There will be no need for input validation
- The last line in the file may or may not have a newline at the end

Expressions

- Consist of simple polynomial terms - the highest degree will be 10
    - If multiple terms include the same exponent, combine those terms
- Exponents will be represented by the ^ character.
- Do not assume that the expression will be in order from highest to lowest exponent.
- All coefficients and exponents will be integers.
    - Exponents may be positive or negative
    - There will not be any exponents of -1
- The | (pipe) character will be used to represent the integral symbol
- If an integral is definite, there will be a number before and after the | character
    - ∫= a|b x dx
    - The endpoints of the interval for the definite integral will be integer values
    - Do not assume a < b
- The variable will always be 'x' and the integral will always end with dx
- There will always be a space before the first term of the integral and before dx
    - Spaces may or may not exist between terms or operators
- If a term has no coefficient, it is assumed to be 1
- **Example Input:**
    - | 3x^2 + 2x + 1 dx
    - 1|4 x^-2 +3x+4 dx
    - -2|2 x^3 – 4x dx

Output:

- All output will be written to the console
- Each anti-derivative will be displayed to a separate line
    - Definite integrals will also include the interval and value (see examples below)
        - Values will be to 3 decimal places
- Use the ^ character to represent exponents
- Order the terms from greatest to least exponent
- Fractions will be simplified
    - All fractions will be enclosed in parentheses
- A space will proceed and follow each operator (+ or -)
- Indefinite anti-derivative format
    - <expression><space><plus><space><capital C><newline>
- Definite anti-derivative format
    - <expression><comma><space><upper bound><pipe><lower bound><space><equal><space><value><newline>

- **Example Output (based on above input):**

    - x^3 + x^2 + x + C
    - (3/2)x^2 + 4x -x^-1, 1|4 = 35.250
    - (1/4)x^4 – 2x^2, -2|2 = 0

EXTRA CREDIT: Add to your program to allow evaluation of indefinite integrals containing trigonometric functions (potential 15 extra points)

- **The trig terms must be stored in the tree with all other nodes.**
- The trig functions will only be sin or cos
- Simple expressions inside the trig functions
    - The term inside the trig function will not have an exponent
- The trig anti-derivatives will be listed at the end of the expression in the order encountered
- There will be a space between the trig function and the inner coefficient of the trig function
    - This is true for both input and output
- **Example Input:**
    - | sin x + cos x dx
    - | 1 – cos 4x dx
    - | 3x^4 – 6x^2 + 2sin 10x dx
- **Example Output:**
    - -cos x + sin x + C
    - x – (1/4)sin 4x + C
    - (3/5)x^5 – 2x^3 – (1/5)cos 10x + C