```cpp
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <cmath>
#include <limits>
#include <vector>

using namespace std;

const string GREEN_COLOR = "\033[32m";
const string RED_COLOR = "\033[31m";
const string YELLOW_COLOR = "\033[33m";
const string CYAN_COLOR = "\033[36m";
const string BLUE_COLOR = "\033[34m";
const string RESET_COLOR = "\033[0m";

struct HistoryEntry {
    string type, input, result;
    HistoryEntry(string t, string i, string r) : type(t), input(i), result(r) {}
};

class Converter {
protected:
    double value;
public:
    Converter(double val = 0.0) : value(val) {}
    virtual ~Converter() = default;
    virtual double convert() const = 0;
    virtual string getType() const = 0;
    void displayResult(const string& input, const string& unitInfo) const {
        const int COL_WIDTH = 15;
        double result = convert();
        cout << fixed << setprecision(2);
        ostringstream oss;
        oss << fixed << setprecision(2) << result;
        string resultStr = oss.str();

        cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+\n";
        cout << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Input" + RESET_COLOR
             << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + unitInfo + RESET_COLOR
             << "|" << setw(COL_WIDTH) << left << GREEN_COLOR + resultStr + RESET_COLOR << "|\n";
        cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+\n";
    }
};

class TemperatureConverter : public Converter {
private:
    char fromUnit, toUnit;
public:
    TemperatureConverter(double temp, char from, char to)
        : Converter(temp), fromUnit(toupper(from)), toUnit(toupper(to)) {}

    double convert() const override {
        if (fromUnit == 'C' && toUnit == 'F') return value * 9.0 / 5.0 + 32.0;
        if (fromUnit == 'F' && toUnit == 'C') return (value - 32.0) * 5.0 / 9.0;
        if (fromUnit == toUnit) return value;
        throw runtime_error("Invalid temperature units (use C or F)");
    }
    string getType() const override { return "Temperature"; }
};

class NumberBaseConverter : public Converter {
private:
    string inputValue;
    char fromBase, toBase;

    string convertToBase(long long num, int base) const {
        if (num == 0) return "0";
        string digits = "0123456789ABCDEF";
        string result;
        while (num > 0) {
            result = digits[num % base] + result;
            num /= base;
        }
        return result;
```

```
    }

public:
    NumberBaseConverter(string val, char from, char to)
        : Converter(0), inputValue(val), fromBase(toupper(from)), toBase(toupper(to)) {
        try {
            if (fromBase == 'B') value = static_cast<double>(stoi(val, nullptr, 2)); // Binary
            else if (fromBase == 'D') value = stod(val); // Decimal
            else if (fromBase == 'O') value = static_cast<double>(stoi(val, nullptr, 8)); // Octal
            else if (fromBase == 'H') value = static_cast<double>(stoi(val, nullptr, 16)); //
Hexadecimal
            else throw runtime_error("Invalid source base (use B, D, O, H)");
        } catch (const invalid_argument& e) {
            throw runtime_error("Invalid number format for the specified base");
        } catch (const out_of_range& e) {
            throw runtime_error("Number out of range for conversion");
        }
    }

    double convert() const override {
        long long intValue = static_cast<long long>(value);
        if (toBase == 'B') return stod(convertToBase(intValue, 2));
        if (toBase == 'D') return value;
        if (toBase == 'O') return stod(convertToBase(intValue, 8));
        if (toBase == 'H') return stod(convertToBase(intValue, 16));
        throw runtime_error("Invalid target base (use B, D, O, H)");
    }

    string getResultString() const {
        long long intValue = static_cast<long long>(value);
        if (toBase == 'B') return convertToBase(intValue, 2);
        if (toBase == 'D') return to_string(intValue);
        if (toBase == 'O') return convertToBase(intValue, 8);
        if (toBase == 'H') return convertToBase(intValue, 16);
        throw runtime_error("Invalid target base");
    }

    void displayResult(const string& input, const string& unitInfo) const {
        const int COL_WIDTH = 15;
        string result = getResultString();

        // Fixed line: Corrected to ensure no stray characters
        cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
string(COL_WIDTH, '-') << "+\n";
        cout << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Input" + RESET_COLOR
             << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + unitInfo + RESET_COLOR
             << "|" << setw(COL_WIDTH) << left << GREEN_COLOR + result + RESET_COLOR << "|\n";
        cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
string(COL_WIDTH, '-') << "+\n";
    }

    string getType() const override { return "Number Base"; }
};

class LogarithmicCalculator : public Converter {
private:
    char logType;
public:
    LogarithmicCalculator(double val, char type) : Converter(val), logType(toupper(type)) {}

    double convert() const override {
        if (value <= 0) throw runtime_error("Logarithm undefined for non-positive numbers");
        if (logType == 'L') return log10(value);  // Base 10
        if (logType == 'N') return log(value);    // Natural log (base e)
        if (logType == 'B') return log2(value);   // Base 2
        throw runtime_error("Invalid log type (use L, N, B)");
    }
    string getType() const override { return "Logarithm"; }
};

class CurrencyConverter : public Converter {
private:
    char fromCurrency, toCurrency;
    static constexpr double INR_TO_USD = 0.012;
    static constexpr double USD_TO_INR = 83.33;
    static constexpr double USD_TO_EUR = 0.92;
    static constexpr double USD_TO_GBP = 0.79;
    static constexpr double EUR_TO_USD = 1.09;
```

```
152         static constexpr double GBP_TO_USD = 1.27;
153     public:
154         CurrencyConverter(double amount, char from, char to)
155             : Converter(amount), fromCurrency(toupper(from)), toCurrency(toupper(to)) {}
156
157         double convert() const override {
158             if (fromCurrency == 'I' && toCurrency == 'U') return value * INR_TO_USD;
159             if (fromCurrency == 'U' && toCurrency == 'I') return value * USD_TO_INR;
160             if (fromCurrency == 'U' && toCurrency == 'E') return value * USD_TO_EUR;
161             if (fromCurrency == 'U' && toCurrency == 'G') return value * USD_TO_GBP;
162             if (fromCurrency == 'E' && toCurrency == 'U') return value * EUR_TO_USD;
163             if (fromCurrency == 'G' && toCurrency == 'U') return value * GBP_TO_USD;
164             if (fromCurrency == toCurrency) return value;
165             throw runtime_error("Invalid or unsupported currency (use I, U, E, G)");
166         }
167         string getType() const override { return "Currency"; }
168     };
169
170     class LengthConverter : public Converter {
171     private:
172         char fromUnit, toUnit;
173         static constexpr double M_TO_FT = 3.28084;
174         static constexpr double FT_TO_M = 0.3048;
175     public:
176         LengthConverter(double length, char from, char to)
177             : Converter(length), fromUnit(toupper(from)), toUnit(toupper(to)) {}
178
179         double convert() const override {
180             if (fromUnit == 'M' && toUnit == 'F') return value * M_TO_FT;
181             if (fromUnit == 'F' && toUnit == 'M') return value * FT_TO_M;
182             if (fromUnit == toUnit) return value;
183             throw runtime_error("Invalid length units (use M or F)");
184         }
185         string getType() const override { return "Length"; }
186     };
187
188     class Calculator {
189     private:
190         double num1, num2;
191         char operation;
192     public:
193         Calculator(double n1, char op, double n2 = 0.0) : num1(n1), num2(n2), operation(toupper(op)) {}
194
195         double calculate() const {
196             switch (operation) {
197                 case '+': return num1 + num2;
198                 case '-': return num1 - num2;
199                 case '*': return num1 * num2;
200                 case '/':
201                     if (num2 == 0) throw runtime_error("Division by zero");
202                     return num1 / num2;
203                 case '^': return pow(num1, num2);
204                 case 'S': return sin(num1 * M_PI / 180.0);
205                 case 'C': return cos(num1 * M_PI / 180.0);
206                 case 'T':
207                     if (cos(num1 * M_PI / 180.0) == 0) throw runtime_error("Tan undefined");
208                     return tan(num1 * M_PI / 180.0);
209                 default: throw runtime_error("Invalid operation (use +, -, *, /, ^, S, C, T)");
210             }
211         }
212
213         void displayResult() const {
214             try {
215                 const int COL_WIDTH = 15;
216                 double result = calculate();
217                 cout << fixed << setprecision(2);
218                 ostringstream oss;
219                 oss << fixed << setprecision(2) << num1 << " " << operation << (operation == 'S' ||
    operation == 'C' || operation == 'T' ? "" : " " + to_string(num2));
220                 string inputStr = oss.str();
221                 oss.str(""); oss << fixed << setprecision(2) << result;
222                 string resultStr = oss.str();
223
224                 cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
    string(COL_WIDTH, '-') << "+\n";
225                 cout << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Input" + RESET_COLOR
226                      << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + inputStr + RESET_COLOR
227                      << "|" << setw(COL_WIDTH) << left << GREEN_COLOR + resultStr + RESET_COLOR <<
```

```
    "|\n";
228             cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
    string(COL_WIDTH, '-') << "+\n";
229         } catch (const runtime_error& e) {
230             const int COL_WIDTH = 45;
231             cout << "+" << string(COL_WIDTH, '-') << "+\n";
232             cout << "|" << setw(COL_WIDTH) << left << RED_COLOR + string("Error: ") + e.what() +
    RESET_COLOR << "|\n";
233             cout << "+" << string(COL_WIDTH, '-') << "+\n";
234         }
235     }
236 };
237
238 class Program {
239 private:
240     vector<HistoryEntry> history;
241
242     void showWelcome() const {
243         const int COL_WIDTH = 40;
244         cout << "+" << string(COL_WIDTH, '-') << "+\n";
245         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Welcome to the Professional
    Converter!" + RESET_COLOR << "|\n";
246         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Advanced conversion and calculation
    tool" + RESET_COLOR << "|\n";
247         cout << "+" << string(COL_WIDTH, '-') << "+\n";
248         // Adding group credits
249         cout << "\n+" << string(COL_WIDTH, '-') << "+\n";
250         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Group Leader: Shravan Nadkarni,
    N-61" + RESET_COLOR << "|\n";
251         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Group Members:" + RESET_COLOR <<
    "|\n";
252         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Parth Ghodke, N-20" + RESET_COLOR
    << "|\n";
253         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Prathamesh Chaumwal, I-31" +
    RESET_COLOR << "|\n";
254         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Gokul Krishnan A V, I-14" +
    RESET_COLOR << "|\n";
255         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Tapas Pandita, N-66" +
    RESET_COLOR << "|\n";
256         cout << "|" << setw(COL_WIDTH) << left << YELLOW_COLOR + "Tejas Waghmare, N-72" +
    RESET_COLOR << "|\n";
257         cout << "+" << string(COL_WIDTH, '-') << "+\n";
258     }
259
260     void clearInputBuffer() const {
261         cin.clear();
262         cin.ignore(numeric_limits<streamsize>::max(), '\n');
263     }
264
265     double getDoubleInput(const string& prompt) const {
266         double input;
267         cout << YELLOW_COLOR << prompt << RESET_COLOR;
268         while (!(cin >> input)) {
269             cout << RED_COLOR << "Invalid input. " << RESET_COLOR << YELLOW_COLOR << prompt <<
    RESET_COLOR;
270             clearInputBuffer();
271         }
272         return input;
273     }
274
275     char getCharInput(const string& prompt) const {
276         char input;
277         cout << YELLOW_COLOR << prompt << RESET_COLOR;
278         cin >> input;
279         clearInputBuffer();
280         return input;
281     }
282
283     string getStringInput(const string& prompt) const {
284         string input;
285         cout << YELLOW_COLOR << prompt << RESET_COLOR;
286         cin >> input;
287         clearInputBuffer();
288         return input;
289     }
290
291     void displayMenu() const {
292         const int COL_WIDTH = 25;
```

```
293         cout << "\n+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+\n";
294         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Option" + RESET_COLOR
295              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Description" + RESET_COLOR << "|\n";
296         cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+\n";
297         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "1" + RESET_COLOR
298              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Calculator" + RESET_COLOR << "|\n";
299         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "2" + RESET_COLOR
300              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Temperature (C/F)" + RESET_COLOR <<
    "|\n";
301         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "3" + RESET_COLOR
302              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Number Base (B/D/O/H)" +
    RESET_COLOR << "|\n";
303         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "4" + RESET_COLOR
304              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Logarithm (L/N/B)" + RESET_COLOR <<
    "|\n";
305         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "5" + RESET_COLOR
306              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Currency (I/U/E/G)" + RESET_COLOR
    << "|\n";
307         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "6" + RESET_COLOR
308              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Length (M/F)" + RESET_COLOR << "|\n";
309         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "7" + RESET_COLOR
310              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "View History" + RESET_COLOR << "|\n";
311         cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "8" + RESET_COLOR
312              << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Quit" + RESET_COLOR << "|\n";
313         cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+\n";
314     }
315
316     void displayHistory() const {
317         const int COL_WIDTH = 15;
318         if (history.empty()) {
319             cout << YELLOW_COLOR << "No history available." << RESET_COLOR << endl;
320             return ;
321         }
322         cout << "\n+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
    string(COL_WIDTH, '-') << "+\n";
323         cout << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Type" + RESET_COLOR
324              << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Input" + RESET_COLOR
325              << "|" << setw(COL_WIDTH) << left << BLUE_COLOR + "Result" + RESET_COLOR << "|\n";
326         cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
    string(COL_WIDTH, '-') << "+\n";
327         for (const auto& entry : history) {
328             cout << "|" << setw(COL_WIDTH) << left << entry.type
329                  << "|" << setw(COL_WIDTH) << left << entry.input
330                  << "|" << setw(COL_WIDTH) << left << GREEN_COLOR + entry.result + RESET_COLOR <<
    "|\n";
331         }
332         cout << "+" << string(COL_WIDTH, '-') << "+" << string(COL_WIDTH, '-') << "+" <<
    string(COL_WIDTH, '-') << "+\n";
333     }
334
335 public:
336     void run() {
337         showWelcome();
338
339         while (true) {
340             displayMenu();
341             int choice = static_cast<int>(getDoubleInput("Enter choice (1-8): "));
342             clearInputBuffer();
343
344             try {
345                 switch (choice) {
346                     case 1: {
347                         double num1 = getDoubleInput("Enter first number: ");
348                         char op = getCharInput("Enter operation (+, -, *, /, ^, S(sin), C(cos),
    T(tan)): ");
349                         if (op != 'S' && op != 'C' && op != 'T') {
350                             double num2 = getDoubleInput("Enter second number: ");
351                             Calculator calc(num1, op, num2);
352                             calc.displayResult();
353                             ostringstream oss;
354                             oss << fixed << setprecision(2) << num1 << " " << op << " " << num2;
355                             string inputStr = oss.str();
356                             oss.str(""); oss << fixed << setprecision(2) << calc.calculate();
357                             history.emplace_back("Calculator", inputStr, oss.str());
358                         } else {
359                             Calculator calc(num1, op);
360                             calc.displayResult();
361                             ostringstream oss;
```

```
                            oss << fixed << setprecision(2) << num1 << " " << op;
                            string inputStr = oss.str();
                            oss.str(""); oss << fixed << setprecision(2) << calc.calculate();
                            history.emplace_back("Calculator", inputStr, oss.str());
                        }
                        break;
                    }
                    case 2: {
                        double temp = getDoubleInput("Enter temperature: ");
                        char from = getCharInput("Enter from unit (C or F): ");
                        char to = getCharInput("Enter to unit (C or F): ");
                        TemperatureConverter tempConv(temp, from, to);
                        ostringstream oss;
                        oss << fixed << setprecision(2) << temp;
                        string inputStr = oss.str();
                        tempConv.displayResult(inputStr, string(1, toupper(from)) + " to " +
string(1, toupper(to)));
                        oss.str(""); oss << fixed << setprecision(2) << tempConv.convert();
                        history.emplace_back(tempConv.getType(), inputStr + " " + string(1,
toupper(from)) + " to " + string(1, toupper(to)), oss.str());
                        break;
                    }
                    case 3: {
                        string number = getStringInput("Enter number: ");
                        char from = getCharInput("Enter from base (B(binary), D(decimal), O(octal),
H(hex)): ");
                        char to = getCharInput("Enter to base (B, D, O, H): ");
                        NumberBaseConverter baseConv(number, from, to);
                        baseConv.displayResult(number, string(1, toupper(from)) + " to " +
string(1, toupper(to)));
                        history.emplace_back(baseConv.getType(), number + " " + string(1,
toupper(from)) + " to " + string(1, toupper(to)), baseConv.getResultString());
                        break;
                    }
                    case 4: {
                        double num = getDoubleInput("Enter number: ");
                        char type = getCharInput("Enter log type (L=log10, N=ln, B=log2): ");
                        LogarithmicCalculator logCalc(num, type);
                        ostringstream oss;
                        oss << fixed << setprecision(2) << num;
                        string inputStr = oss.str();
                        string logStr = (type == 'L' ? "log10" : type == 'N' ? "ln" : "log2");
                        logCalc.displayResult(inputStr, logStr);
                        oss.str(""); oss << fixed << setprecision(2) << logCalc.convert();
                        history.emplace_back(logCalc.getType(), logStr + "(" + inputStr + ")",
oss.str());
                        break;
                    }
                    case 5: {
                        double amount = getDoubleInput("Enter amount: ");
                        char from = getCharInput("Enter from currency (I, U, E, or G): ");
                        char to = getCharInput("Enter to currency (I, U, E, or G): ");
                        CurrencyConverter currConv(amount, from, to);
                        ostringstream oss;
                        oss << fixed << setprecision(2) << amount;
                        string inputStr = oss.str();
                        currConv.displayResult(inputStr, string(1, toupper(from)) + " to " +
string(1, toupper(to)));
                            oss.str(""); oss << fixed << setprecision(2) << currConv.convert();
                        history.emplace_back(currConv.getType(), inputStr + " " + string(1,
toupper(from)) + " to " + string(1, toupper(to)), oss.str());
                        break;
                    }
                    case 6: {
                        double length = getDoubleInput("Enter length: ");
                        char from = getCharInput("Enter from unit (M or F): ");
                        char to = getCharInput("Enter to unit (M or F): ");
                        LengthConverter lenConv(length, from, to);
                        ostringstream oss;
                        oss << fixed << setprecision(2) << length;
                        string inputStr = oss.str();
                        lenConv.displayResult(inputStr, string(1, toupper(from)) + " to " +
string(1, toupper(to)));
                        oss.str(""); oss << fixed << setprecision(2) << lenConv.convert();
                        history.emplace_back(lenConv.getType(), inputStr + " " + string(1,
toupper(from)) + " to " + string(1, toupper(to)), oss.str());
                        break;
                    }
```

```
                    case 7:
                        displayHistory();
                        break;
                    case 8: {
                        const int COL_WIDTH = 40;
                        cout << "+" << string(COL_WIDTH, '-') << "+\n";
                        cout << "|" << setw(COL_WIDTH) << left << CYAN_COLOR + "Thank you for using Professional Converter!" + RESET_COLOR << "|\n";
                        cout << "+" << string(COL_WIDTH, '-') << "+\n";
                        return;
                    }
                    default:
                        const int COL_WIDTH = 40;
                        cout << "+" << string(COL_WIDTH, '-') << "+\n";
                        cout << "|" << setw(COL_WIDTH) << left << RED_COLOR + "Invalid choice. Please select 1-8." + RESET_COLOR << "|\n";
                        cout << "+" << string(COL_WIDTH, '-') << "+\n";
                }
            } catch (const runtime_error& e) {
                const int COL_WIDTH = 45;
                cout << "+" << string(COL_WIDTH, '-') << "+\n";
                cout << "|" << setw(COL_WIDTH) << left << RED_COLOR + string("Error: ") + e.what() + RESET_COLOR << "|\n";
                cout << "+" << string(COL_WIDTH, '-') << "+\n";
            }
        }
    }
};

int main() {
    Program program;
    program.run();
    return 0;
}
```