**REPORT FOR LAB1**

**I. Các hàm được xây dựng trong bài**

string f\_minus(string s1, string s2);

string f\_add(string s1, string s2);

string f\_mul(string s1, string s2);

string f\_div(string s1, string s2);

bool isNegative(string s);

void deleteBackspace(string &line);

bool isDigit(char c);

bool isOp(string s);

vector<string> getElements(string line);

int priority(string op);

vector<string> makeOrder(string line);

string process(string line);

void process\_file(string input, string output);

Usage:

g++ 24120036.cpp -o 24120036

./24120036 tests.txt output\_24120036.txt

- isNegative(string s) tự code 100%

logic: trả về value ký tự đầu tiên của chuỗi có dấu trừ hay không

- f\_minus(string s1, string s2) code 80% AI fix bug 20%(chỉnh sữa cách mượn số khi trừ) hiểu 100% logic

logic:

+ Khâu xử lí chính: so sánh 2 chuỗi số để quyết định xem kết quả của chúng là âm hay dương thông qua biến bool negative. Duyệt tường ký tự trong 2 chuỗi số đến khi cả 2 đều được duyệt hết, khi duyệt quy đổi từ char sang int bằng phương pháp - '0' để sang int và từ int + '0' để về lại char. Sau đó xử lý phép trừ bằng cách trừ từng digit từ sau chuỗi đến đầu chuỗi bằng logic của phép trừ thông qua biến borrow để mượn khi trừ digit ra âm, thông qua borrow ta xử lý digit của s1 trước bằng s1\_digit -= borrow. So sánh digit s1 và s2 nếu bé hơn thì tăng 10 đơn vị cho s1 rồi lấy s1\_digit - s2\_digit rồi đặt biến borrow = 1 vì đã mượn 10 còn nều ngược lại thì không cần và đặt borrow = 0 rồi chèn kết quả minus\_digit và string kết quả r. Xử lý các phần '0' dư thừa trong những trường hợp 0002 - 0001 = 0001. Trước khi trả về thì kiểm tra điều kiện xem biến bool ở đầu hàm chúng ta xử lý có âm hay không để thêm dấu '-' vào kết quả r.

+ Chia ra các trường hợp ở đầu hàm để trả về đúng kết quả ví dụ:

if (isNegative(s1) && isNegative(s2)) {

     s1.erase(0, 1);

     s2.erase(0, 1);

     return f\_minus(s2, s1);

}

Khi lấy 2 số âm trừ nhau -a - (-b) thì bằng b - a.

- f\_add(string s1, string s2) tự code 100% hiểu 100%

+ Đầu tiên cũng xử lý các điều kiện cơ bản như hàm f\_minus

+ Khâu xử lý chính: cũng lấy digit đến hết như ở hàm trên nhưng ở khâu tính toán không cần kiểm tra s1\_digit và s2\_digit cái nào lớn hơn mà chỉ cần cộng thẳng 2 digit rồi % ra, bonus thì lấy kết quả vừa rồi chia lấy nguyên là sẽ được lượng được cộng thêm. Chèn add\_digit vừa mới tính vào hàm kết quả r rồi xử lý các số '0' dư thừa.

- f\_mul(string s1, string s2)

+ Kiểm tra 1 trong 2 hay 2 có phải là "0" để trả về ngay "0"

+ Thêm biến bool negative như f\_minus nhưng kiểm tra khác theo logic của phép nhân

+ Lấy từng digit của s1 rồi nhân với tất cả digit của s2 thông qua 2 for, kết quả này được lưu vào mảng res với kích thước của res tối đa = size của 2 string theo logic của phép nhân. Sau khi chạy xong thì thu được mảng res với từng phần tử của res là từng kết quả int của từng digit s1 nhân với từng digit s2, xử lý biến nhớ carry cho res.

+ Chuyển thành string rồi insert vào r và thêm dấu nếu cần.

- f\_div(string s1, string s2)

+ Kiểm tra số chia nếu = 0 thì báo lỗi ERROR, số bị chia s1 = 0 thì trả kết quả là 0

+ Sử dụng biến bool negative như hàm nhân với điều kiệm kiểm tra tương tự

+ Thực hiện phép chia lấy phần nguyên: Lặp qua từng chữ số của số bị chia,

xây dựng số bị chia tạm thời (current) và đếm số lần trừ được số chia từ số bị chia tạm thời.

Sau đó ghi số lần trừ được vào kết quả.

+ Loại bỏ các số 0 vô nghĩa ở đầu chuỗi kết quả.

+ Thêm dấu âm nếu cần.

bool isNegative(string s): Tự code 100%.

Logic: Trả về true nếu chuỗi s bắt đầu bằng dấu '-', ngược lại trả về false.

Hàm này kiểm tra xem số s có âm không bằng cách xem ký tự đầu tiên có phải là dấu '-' không.

void deleteBackspace(string &line): Tự code 100%.

Logic: Loại bỏ tất cả khoảng trắng trong chuỗi line. Duyệt qua từng ký tự của chuỗi, nếu ký tự đó không phải là khoảng trắng (' '), thì giữ lại ký tự đó.

Hàm này duyệt chuỗi line và loại bỏ mọi ký tự khoảng trắng.

bool isDigit(char c): Tự code 100%.

Logic: Kiểm tra xem ký tự c có phải là chữ số ('0' đến '9') không. Trả về true nếu c là một chữ số, ngược lại trả về false.

Hàm này kiểm tra xem một ký tự có phải là số không.

bool isOp(string s): Tự code 100%.

Logic: Kiểm tra xem chuỗi s có phải là một trong các toán tử "+", "-", "\*", "/", "(", ")" không. Trả về true nếu s là một toán tử, ngược lại trả về false.

Hàm này kiểm tra xem một chuỗi có phải là một toán tử không.

vector<string> getElements(string line): Tự code 100%.

Logic: Phân tích chuỗi line thành các phần tử (số hoặc toán tử) và lưu vào một vector các chuỗi.

Duyệt qua chuỗi line, xác định xem mỗi ký tự là một phần của số, toán tử, hay dấu ngoặc.

Thêm mỗi phần tử vào vector kết quả.

Ví dụ: Nếu line là "1 + 2 \* 3", hàm sẽ trả về vector {"1", "+", "2", "\*", "3"}.

int priority(string op): Tự code 100%.

Logic: Trả về độ ưu tiên của toán tử op (ví dụ: "\*" và "/" có độ ưu tiên cao hơn "+" và "-"). Sử dụng câu lệnh if-else if-else để xác định độ ưu tiên dựa trên giá trị của op.

Hàm này trả về mức độ ưu tiên của một toán tử. Toán tử có độ ưu tiên cao hơn sẽ được thực hiện trước.

vector<string> makeOrder(string line): Tự code 100%.

Logic: Chuyển đổi biểu thức toán học trong chuỗi line sang dạng ký pháp Ba Lan ngược (Reverse Polish Notation - RPN) để dễ dàng tính toán.

Sử dụng một stack để lưu trữ các toán tử và dấu ngoặc.

Duyệt qua các phần tử của biểu thức, thêm số vào output, và xử lý các toán tử dựa trên độ ưu tiên của chúng.

Ví dụ: Nếu line là "1 + 2 \* 3", hàm sẽ trả về vector {"1", "2", "3", "\*", "+"}.

string process(string line): Tự code 100%.

Logic: Tính toán giá trị của biểu thức toán học được biểu diễn dưới dạng chuỗi line (đã được chuyển đổi sang RPN).

Sử dụng một stack để lưu trữ các toán hạng.

Duyệt qua các phần tử RPN, đẩy số vào stack, và thực hiện các phép toán khi gặp toán tử.

Ví dụ: Nếu line là "1 2 3 \* +", hàm sẽ trả về "7".

void process\_file(string input, string output): Tự code 100%.

Logic: Đọc các biểu thức toán học từ file input, tính toán giá trị của từng biểu thức và ghi kết quả vào file output.

Mở file input và output.

Đọc từng dòng từ file input, coi mỗi dòng là một biểu thức toán học.

Gọi hàm process để tính toán giá trị của biểu thức.

Ghi kết quả vào file output.