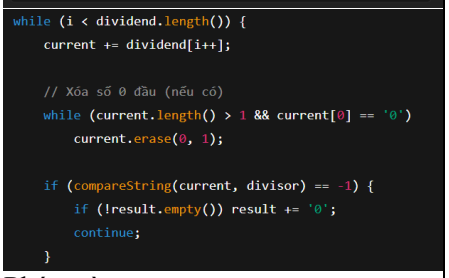
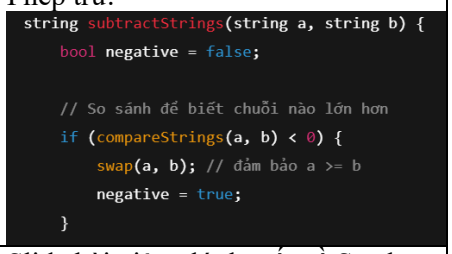
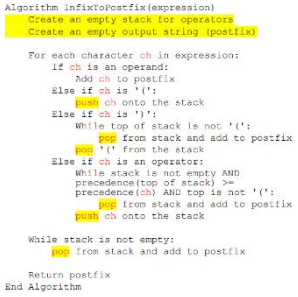
REPORT LAB 1

I. Thông tin chung:

Họ và tên: Lê Công Anh Khoa
Mã số sinh viên: 24120072

No.	Percentage understood	Content understood	Percentage Referenced	Content Referenced	Reference Source
1. Triển khai Stack kiểu char	100%	Nguyên lý hoạt động của Stack và cách tự tạo cấu trúc dữ liệu Stack.	0%	Không có	Tinh chỉnh lại code từ bài tập tuần 5.
2. Xử lý số nguyên lớn	100%	- Sử dụng string để thực hiện các phép toán +, -, *, / với dữ liệu là các số nguyên lớn. - Xử lý trường hợp phép trừ ra số âm. - Xử lý phép chia số 0.	20%	**- Đối với phép chia:** phương pháp tách chuỗi bị chia thành các "khối" (lớn hơn hoặc bằng số chia), rồi chia theo từng khối để giảm thời gian thực hiện dù đầu vào lớn. **- Phép trừ chuỗi:** thêm cờ negative để đánh giá kết quả ra âm, so sánh chuỗi trước khi trừ để xét trường hợp kết quả âm. (Tìm kiếm hướng giải quyết và tự code lại theo cách hiểu)	*Chat GPT.* Phép chia:   Phép trừ: 
3. Chuyển phép tính từ dạng infix thành postfix	100%	- Sử dụng vector<string> để lưu dạng postfix và Stack để lưu tạm thời các toán tử và dấu '('. - Dựa vào độ ưu tiên mà đẩy toán tử từ Stack vào vector postfix. - Xử lý các trường hợp lỗi cú pháp: Thừa toán tử, thừa\thiếu dấu ngoặc đơn ( ), toán tử cạnh nhau...	0%	Dùng stack để lưu tạm thời các toán tử và phân chia độ ưu tiên của toán tử và dấu ( ). (Tham khảo pseudo code)	Slide bài giảng lý thuyết về Stack 

4. Xử lý dạng postfix và đưa ra kết quả của bài toán	100%	- Từ dạng postfix của phép tính, đọc vào các toán hạng và sử dụng chúng để thực hiện các toán tử. - Xử lý được các trường hợp số âm và thừa, thiếu toán tử/toán hạng.	0%	Không có	

(Tất cả các nội dung đều là tự code và chỉ tham khảo hướng giải quyết từ các nguồn đã kể trên)

II. Những điều đã làm để hoàn thành bài tập:

1. Xác định nội dung cần thực hiện:

- Chia bài làm thành các nội dung lớn gồm:
  - Triển khai kiểu dữ liệu Stack lưu trữ các char.
  - Triển khai các hàm xử lý phép toán số nguyên lớn, lưu trữ dưới dạng string.
  - Phân tích và thực hiện cú pháp từ các phép toán đầu vào.

2. Thực hiện lần lượt các nội dung:

a) Triển khai kiểu dữ liệu Stack lưu trữ các char:
   - Dựa vào các hàm đã khai báo từ bài tập tuần 5 đối Stack theo dữ liệu int và sửa lại thành dữ liệu char.

b) Triển khai các hàm xử lý phép toán số nguyên lớn, lưu trữ dưới dạng string:
   - Phép so sánh 2 chuỗi a và b: so sánh độ dài trước sau đó đến so sánh từng ký tự của chuỗi, trả về -1 nếu a<b, 0 nếu a=b, 1 nếu a>b.
   - Các phép toán +, -, * thực hiện từng chữ số một như cách làm khi tính bằng giấy.
   - Với phép cộng: thì phải lưu trữ phần thừa 1 với mỗi chữ số.
   - Với phép trừ: so sánh 2 chuỗi trước khi thực hiện và đổi chỗ nếu cần để bài toán luôn ở dạng số lớn hơn trừ số bé hơn, đánh dấu negative nếu số trừ lớn hơn số bị trừ.
   - Với phép nhân: đổi chỗ để bài toán a*b luôn là b ngắn hơn a, nhân từng chữ số của b từ phải qua trái với a rồi cộng các kết quả lại, cộng lệch đi 1 ký tự về bên trái với mỗi chữ số của b về bên trái.
   - Với phép chia:

        - Vì là chia lấy phần nguyên nên ta chỉ chia nếu số bị chia lớn hơn số chia.
        - Để không bị vượt quá thời gian với các phép chia từ 100 số, ta phải tách số bị chia thành từng khối: thực hiện lấy các số từ trái qua phải của số bị chia để nhóm thành khối cur (cur>=số chia) rồi chia cur với số chia.
        - Vì không thể chia trực tiếp nên ta tìm giá trị count lớn nhất sao cho tích của count với số chia bé hơn cur, sau đó trừ cur cho tích ấy rồi tiếp tục tách khối, cộng các count có được vào res. Thêm số 0 sau res với mỗi bước mà ta thêm 1 số mớ vào cur mà cur vẫn bé hơn số chia.

    c) Phân tích và thực hiện cú pháp từ các phép toán đầu vào:

- *Hàm chuyển từ infix thành postfix:*
    - lưu trữ postfix dưới dạng vector<string>, lưu trữ các toán tử tạm thời bằng Stack.
    - Duyệt qua từng ký tự của infix, chia ra thành các trường hợp: kí tự trắng, số, dấu ngoặc ( ), toán tử và các ký tự ngoài.
    - Gặp kí tự trắng: bỏ qua.
    - Gặp số: tiến hành lấy từng số bỏ vào string và đẩy string đó vào postfix.
    - Sau dấu mở ngoặc ‘(’ thì có 3 trường hợp:
        - Dấu ‘-’: biểu thị số âm.
        - Số: trong dấu ngoặc là 1 phép toán con.
        - Dấu mở ngoặc ‘(’: các dấu ngoặc lồng nhau.
    - Gặp dấu đóng ngoặc ‘)’ thì phải lấy hết các toán tử còn lại giữa 2 dấu ngoặc cùng cấp bỏ vào postfix.
    - Gặp toán tử: tiến hành so sánh độ ưu tiên của toán tử hiện tại và toán tử trên cùng của Stack, nếu Stack không rỗng thì ta lấy lần lượt tất cả các toán tử có độ ưu tiên lớn hơn hoặc bằng toán tử đang xét và bỏ vào postfix theo thứ tự từ trên xuống của Stack.
    - Ký tự có độ ưu tiên = 0 và không thuộc các trường hợp trên là ký tự không xác định, trong trường hợp đó trả về vector rỗng (lỗi cú pháp).

- Trong hàm còn xử lý thêm các trường hợp lỗi cú pháp có thể gặp phải (trả về vector rỗng nếu có lỗi cú pháp), xóa các số 0 ở đầu của toán hạng trước khi thêm vào postfix.

- *Hàm tính toán dựa trên postfix:*
  - Kiểm tra xem vector postfix có rỗng không, nếu rỗng nghĩa là đã vi phạm lỗi cú pháp đã định nghĩa ở hàm chuyển từ infix thành postfix.
  - Tạo một vector<string> nums để lưu trữ các toán hạng.
  - Lấy từng giá trị của vector postfix:
    - Nếu giá trị đó là toán hạng thì push vào vector nums.
    - Nếu là toán tử thì lấy 2 toán hạng sau cùng được thêm vào vector nums để thực hiện phép tính.
  - Ở bước thực hiện phép tính, thay vì phải tính toán với số âm và dương thì ta đặt biến bool cho từng toán hạng để kiểm tra xem nó có âm không, nếu âm thì ta chỉ lấy phần số của toán hạng (bỏ dấu '-') rồi thực hiện phép tính với số dương, sau đó tùy trường hợp mà ta thêm dấu '-' trước kết quả hoặc biến đổi từ phép cộng thành phép trừ cho phù hợp.
  - Đẩy ngược kết quả của từng phép tính vào vector nums, sau tất cả thì phần tử cuối cùng còn lại của nums chính là đáp án của phép tính.