REPORT

-the approach used in each solution-

-24120072_Lê Công Anh Khoa-

A. Stack:

1. Stack* initializeStack():

- Tạo con trỏ *s chỉ đến một Stack mới: Stack* s=new Stack();
- Đặt phần tử trên cùng của Stack thành nullptr và trả về con trỏ s

2. void push(Stack &s, int key):

- Tạo một NODE mới và gán vào con trỏ newNode.
- Đặt giá trị key của newNode thành giá trị key truyền vào hàm.
- p_next của newNode đặt bằng nullptr.
- Kiểm tra nếu Stack rỗng thì phần tử top=newNode.
- Nếu không p_next của newNode=phần tử top.
- Cập nhật top mới=newNode.

3. int pop(Stack &s):

- Kiểm tra nếu Stack rỗng, trả về -1.
- Đặt đặt temp=s.top để lấy NODE trên cùng, val=temp->key để lấy giá trị của NODE đó.
- Cập nhật top=top->p_next sau đó xóa temp, trả về val.

4. int size(Stack s):

- Kiểm tra nếu Stack rỗng, trả về 0.
- Tạo biến đếm count=0.
- Duyệt phần tử của Stack từ top đến hết, mỗi lần như thế tăng biến count lên 1.
- Trả về biến count.

5. bool isEmpty(Stack s):

- Kiểm tra phần tử trên cùng của Stack có bằng nullptr không, nếu có thì trả về true, stack rỗng và ngược lại.

6. Hàm main:

- Để có thể đọc được dữ liệu từ file input và ghi ra file output, ta thực hiện mở 2 file input.txt và output.txt.
- Thực hiện đọc từng dòng của file input.txt đến khi hết file.
- Kiểm tra nếu dòng vừa đọc rỗng hoặc bắt đầu bằng ký tự // thì bỏ qua dòng đó và tiếp tục đọc.
- Sử dụng stringstream để đọc từng chữ của dòng.

- Đọc chữ đầu tiên vào biến action kiểu string, là hành động người dùng muốn thực hiện.
- Dựa vào action mà thực hiện các hàm như đã khai báo ở trên.
- Lưu ý một số điểm như sau:
    - Nếu action=”init” và Stack đã từng được tạo trước đó thì ta phải giải phóng Stack hiện có và tạo lại Stack rỗng mới.
    - Nếu action=”push” thì ta tiếp tục đọc vào chữ tiếp theo của dòng vào biến val, theo yêu cầu đề thì đây sẽ là giá trị muốn push vào Stack, có kiểu dữ liệu int.
    - Ở mỗi hành động push/pop thì ta kiểm tra xem nếu s=nullptr nghĩa là Stack chưa được tạo thì bỏ qua dòng lệnh hiện tại.
    - Sau mỗi dòng ta thực hiện in trạng thái của Stack, vì theo yêu cầu đề thì Stack phải được in từ bottom tới top nên sử dụng hàm printReverse để in giá trị của Stack từ dưới lên một cách đệ quy.

B. Queue

1. Queue* initializeQueue():

- Tạo Queue trống và gán vào con trỏ *q.
- Đặt q->head=nullptr=q->tail.
- Trả về con trỏ q.

2. void enqueue(Queue &q, int key):

- Tạo NODE rỗng mới và gán vào con trỏ newNode, đặt key của newNode = key là giá trị truyền vào hàm, p_next của newNode=nullptr.
- Nếu Queue rỗng thì gán q.head=q.tail=newNode.
- Nếu không thì q.tail->p_next=newNode. Cập nhật q.tail=newNode.

3. int dequeue(Queue &q):

- Nếu Queue rỗng thì trả về -1.
- Tạo con trỏ temp để lưu q.head của Queue, biến val để lưu giá trị của q.head đó.
- Cập nhật q.head=q.head->p_next, nếu q.head mới bằng nullptr thì cập nhật q.tail=nullptr.
- Xóa temp và trả về val.

4. int size(Queue q):

- Nếu Queue rỗng thì trả về 0.

- Tạo biến đếm count=0 và duyệt qua từ head tới tail của Queue, trả về biến count.

5. bool isEmpty(Queue q):

- Kiểm tra nếu q.head bằng nullptr, Queue rỗng, trả về true và ngược lại.

6. Hàm main:

- Tương tự hàm main của Stack, chỉ thay đổi các điều kiện về action, thay vì push thì sẽ là enqueue và pop thì sẽ là dequeue.
- Lần này chỉ cần in Queue từ head tới tail.