Lab 2, CS201 (2020)

Write a C program to apply the concepts of stacks and queues for addressing these five different types of queries. **Read the assignment carefully.**

- Single .c file. Filename format instructions are mentioned at end.
- Input outputs that you tried or used for testing your code can be included in your program file at end within comments (using /* */). Also mention your name, entry number and self declaration regarding doing assignment yourself (expected). Mention clearly if any weblinks you refer to / whatever assistance you took from others (and from whom ?).

The 5 queries are denoted by **S, Q, I, E,** and **P** and these query type identifiers correspond to -

- performing **S**tack operations,
- performing **Q**ueue operations,
- converting **I**nfix expression to equivalent postfix expression,
- **E**valuating infix expression value and
- computing stack **P**ermutation, respectively.

The first line of the input contains an integer **T** and **Z** where **T** denotes the number of queries and **Z** denotes the size for the stack and queue (wherever that size limit is applicable).
Then there follows T lines, each line correspond to a query and query type is identified with the first character in that line which could be either **S, Q, I, E** or **P.**

Query type identifier is then followed by parameters specific to that query as discussed below. These parameters would be single-space separated.

Query Type	Parameter	Functionality / Response / Remarks
S	Integers	// Any valid integer
	Positive Integer	Push that +ve integer value in stack. Don't print anything.
	0	Is_Stack_Empty. Print E if yes, else NE
	-1	Pop. Print the integer value popped out. If nothing to pop, print E.
	-2	Is_Stack_Full. Print F if yes, else NF
	-3	Print the number of elements currently in stack.
	Any other integer	Denotes end of the query. Ignore any symbol after this in that sequence
		//Note: Responses are to be single space-separated.
Q	Integers	//Any valid integer
	Positive Integer	Insert/Enqueue that value in the queue. Don't print anything except if queue is Array based and is full. If full and can't enqueue, print F
	0	Is_Queue_Empty. Print E if yes, else NE
	-1	Dequeue. Print the value you dequeued, If nothing to delete, print E

<table>
<tr><td></td><td>-2
-3

Any other integer</td><td>Is_Queue_Full. Print F if yes, else NF
Print the Number of elements currently in queue

Denotes end of the query. Ignore any symbol after this

Query Response:

//Note: Responses are to be single space-separated.</td></tr>
<tr><td>I</td><td>Integers, Operators, #</td><td>Max 20 operators.
used to mark the end of query. Ignore any symbol after
Integers range : 0 to 1000
No brackets etc. No operators other than what mentioned below would be considered valid.
Operands and Operators are single-space separated.
Some operators may consist of two characters e.g. >>
Query Response would be:
Postfix expression where operands and operator are single-space separated. No space before first operand.
If expression is invalid, mention ONLY “Error” without Quotes</td></tr>
<tr><td>E</td><td>Integers, Operators, #</td><td>Max 20 operators.
used to mark the end of query. Ignore any symbol after #.
Integers range : -1000 to 1000
No brackets etc. No operators other than what mentioned below would be considered valid.

<table><tr><th>Operators</th><th>Associativity</th></tr><tr><td>* / %</td><td>left to right</td></tr><tr><td>+ -</td><td>left to right</td></tr><tr><td><< >></td><td>left to right</td></tr><tr><td>< <= > >=</td><td>left to right</td></tr><tr><td>== !=</td><td>left to right</td></tr><tr><td>&</td><td>left to right</td></tr><tr><td>^</td><td>left to right</td></tr><tr><td>|</td><td>left to right</td></tr></table>
Operands and Operators are single-space separated.
Some operators may consist of two characters e.g. >> (there is no space between these characters)
Note: &, ^, | are bitwise operators

<u>Query Response:</u>
Value of the evaluated expression
If expression was invalid, mention “Error” without quotes</td></tr>
<tr><td>P</td><td>N k</td><td>k^{th} sequence (in lexicographic ordering) in stack permutations possible using N numbers observed in the order as follows: 1,2,3,…,N. (N would be less than equal to 30)
<u>Query Response:</u>
Total number of stack permutations possible with these N different numbers, and then the k^{th} stack permutation.</td></tr>
</table>

Stack for integers to be implemented using linked list. Stack for non integer elements to be implemented using array.
Queue (if to be implemented) can be implemented using either array or linked-list.
You can use string.h (#incldue <string.h>) if you so need.
All queries or say all rows are independent. No query dependent on previous query etc.

Print **Error** if query type identifier is other than what's discussed above.

Print **Error** if parameters are not specific to the query type under consideration or parameters are not in proper format as mentioned above.

Sample Input1:	**Sample Output1**
7 25 S 6 20 30 4 -1 -1 13 16 -2 -3 0 -2 -4 S -1 6 2 10 30 14 -1 -1 13 16 -2 -3 0 -2 -4 Q 16 20 30 4 -1 -1 13 16 -2 -3 0 -2 -4 1 2 3 -1 I 1 + 2 * 3 # I 1+ + 2 # I 2 * + + 3 # P 2 2	4 30 NF 4 NE NF E 14 30 NF 5 NE NF 16 20 NF 4 NE NF 1 2 3 * + Error Error 2 2 1
Sample Input2: 7 30 E 1 + 2 * 3 # E 1 + 3 – 5 # E 4 + 3 & 5 <= 18 # E 4 + 3 * 5 <= 18 # P 3 2 P 3 5 I 1+ 2 * #	**Sample Output2** 7 -1 1 0 5 1 3 2 5 3 2 1 Error

Explanation for the Example Input1/Output1:

There would be 5 queries. Stack /Queue size is 20.

Next there are 5 queries: One for stack, then One for queue and then 2 for infix2postfix and at last, one query for Stack Permutation. They are self explanatory.

Weightage: S and Q : 10% each , I and E : 20% each, P : 40 %
I / E : 10 % only if your program don't support 2 characters operators e.g. >=

Note: Refer to next page also.

You MUST follow the Filename format: L02_<Your First name>_<Entrynumber>_CS201_2020.c For e.g. L02_Raman_2019CSB9999_CS201_2020.c for

The form allows one to submit more than once. However note that total 5 attempts only permitted for the same, and for any 2nd / 3rd /... resubmission, you must mention suffix _2 or _3 (as appropriate) while submitting your file. For e.g.
2nd attempt would be L02_Raman_2019CSB9999_CS201_2020_2.c
and fifth attempt would be L02_Raman_2019CSB9999_CS201_2020_5.c

If you could not implement successfully some query or implemented it for partial cases only, do mention clearly about that in your program in comments section and also in the google form once while submitting.

Form for submission will be provided later next week on Wednesday before the due date i.e. Sep 18, 2020 11:59:59pm
For late submissions:
Upto 15 mins late - Concession, no penalty
15mins to 1hour late: 5%
1- 2hours late: 10%
Similarly 5% penalty for every 1 hour late. for upto 10 hours i.e. upto 10am of the next day.
After 10am and before 4pm: 60% penalty
After 5pm and before 10pm: 70% penalty
After 10pm of next day and before midnight: 80% penalty
After 24 hours: 100% penalty

Following are some queries that I received.

Q1) For query S, do we have to implement stack using array or using linked list. If we have to use linked list, do we have to implement Is_Stack_Full using max size 'Z'.
Ans:
Assignment clearly mentions:
The first line of the input contains an integer T and Z where T denotes the number of queries and Z denotes the size for the stack and queue (wherever that size limit is applicable).
If parameter -3, i.e. Is_Stack_Full. Print F if yes, else NF
Stack for integers to be implemented using linked list. Stack for non integer elements to be implemented using array.

So this implies that ==> Yes, you have to implement "Is_Stack_Full"
And as Stack is of integers and implemented using linked list, answer would be NF whenever the query comes

Q2) Can we assume that a query will always end i.e. for example for query S, -4 will always be there?
Ans:
Yes, you can always assume that. -4 or -5 or -6 or any other integer less than -3.
Also, if you see assignment, it mentions clearly about the same.

Q3) Can we assume that inputs/parameters corresponding to a query won't be corresponding to another query. For example, for query S, we can't have parameters corresponding to query I.
Ans: Yes you can assume that.
The parameters following the query S and Q would always be integers. no #, +, character symbols etc would be there. I understand that there wouldbe challenge in reading otherwise. You are using scanf("%d"..) and not therefore expecting characters. We won't have any weird test case of having char for S and Q

As such we do mention : Print "Error" if parameters are not specific to the query type under consideration or parameters are not in proper format as mentioned above.
This is mention specifically in reference to queries E and I. but therefore we will be as such restricting our test cases with parameters from the set of options mentioned in column 2. Integers, Operators, and # only. And list of operators is already provided clearly.