ALGORITMA – STACK AND QUEUE

1. Buat kelas QueueStack dilanjutkan dengan method __init__stack dan queue yang kosong.

2. Buat method push_front untuk menambahkan data ke dalam stack.

3. Buat method push_back untuk menambahkan data ke dalam queue.

4. Buat method pop_front() untuk menghapus dan mengembalikan elemen paling atas/depan dari stack / queue sesuai aturan FIFO.

5. Jika stack kosong, balikkan urutan elemen dalam queue dan tambahkan ke stack.

6. Balikkan elemen teratas dari stack.

7. Jika queue kosong, balikkan urutan elemen dalam stack dan tambahkan ke queue.

8. Balikkan elemen teratas dari queue.

9. Buat metode pop_back untuk menghapus dan mengembalikan elemen terakhir dari stack atau queue sesuai aturan LIFO.

10. Jika queue kosong, balikkan urutan elemen dalam stack dan tambahkan ke queue.

11. Balikkan elemen terakhir dari queue.

12. Jika stack kosong, balikkan urutan elemen dalam queue dan tambahkan ke stack.

13. Balikkan elemen terakhir dari stack.

14. Buat method __str__() untuk mengembalikan kondisi terakhir QueueStack.

15. Dalam metode ini, gabungkan elemen stack dan queue secara terbalik.

16. Dalam perulangan while True, baca jumlahPerintah dari masukan pengguna.

17. Inisialisasi objek qs sebagai kelas QueueStack.

18. Dalam perulangan for, pengguna memasukkan perintah ke variable “perintah”.

19. Untuk perintah 'push back' atau 'push front', tambahkan nilai yang dimasukkan ke dalam stack atau queue.

20. Untuk perintah 'pop back' atau 'pop front', lakukan operasi pop sesuai aturan FIFO atau LIFO.

21. Cetak kondisi terakhir dari QueueStack, diikuti inisial qs.

22. Tanya kepada pengguna apakah mereka ingin melanjutkan atau tidak.

23. Jika jawabannya = "tidak", keluar dari perulangan while. Jika “ya” = lanjut kembali.

24. Cetak pesan program selesai.