i) CPP Program to implement Queue using two stack with costly enqueue().

→ Initalize Queue 'q' is created and both stack 'S1' and 'S2' are initially empty.

→ Enqueue 1: Push 1 onto 'S1'
S1: 1
S2:

→ Enqueue 2: Move all elements from 'S1' to 'S2' Push 2 onto 'S1' and then move everything back to 'S1'.
S1: 2 1
S2:

→ Enqueue 3: Move all elements from 'S1' to 'S2' Push '3' onto 'S1' and then move everything back to 'S1'
S1: 3 2 1
S2:

→ Dequeue: Return the Top of 'S1' which is 1 and Pop it
S1: 3 2
S2:

→ Dequeue: Return the Top of 'S1' which is 2 and Pop it
S1: 3
S2:

→ Dequeue: Return the Top of S1 which is 3 and Pop it
S1:
S2:

Output of this is
1
2
3

CPP Program to implement Queue using two Stack with Costly dequeue().

→ Initialize Queue: qu is created and both stack S1 and S2 are initally empty.

→ Dequeue from empty Queue: Try to dequeue but both S1 and S2 are empty. So return -1.

→ Enqueue: Push 1 onto S1
S1: 1
S2:

→ Dequeue: check if S2 is empty move elements from S1 to S2 then return the top of S2 and Pop it
S1:
S2: 1

→ The output is -1
1

CPP Dequeue.

→ Initialize dequeue: dequeue is an array of size 100 front and rear are initialized to -1

→ Display menu
1) Insert at the front end of Dequeue

2) Insert at the rear end of Deque.
3) Delete from the front end of Deque.
4) Deleted from the rear end of deque
5) Display
6) Exit.

3) user input.

Enter choice: 1

Value: 10

→ Insert at the front End

Insert 10 at the front of deque.

→ user input

Enter choice: 2

Value 20.

→ Insert at the rear END

Deque case: 10 20

→ user input

choice: 3

→ Delete from the front

Deleted element is 10

→ user input

choice: 5

→ Display deque

Deque: 20.

Priority cpp

→ Initalize priority: queue is created, and the front pointer is initially set to nullptr.

→ Enqueue operation.

Enqueue(3,3);
Enqueue(1,1)
Enqueue(2,2);
Enqueue(4,4).

Queue: (1,1) → (2,2) → (3,3) → (4,4)

→ Display priority Queue.

Data: 1, Priority: 1
Data: 2, Priority: 2
Data: 3, Priority: 3
Data: 4, Priority: 4.

→ Dequeue operation

Dequeue.
Dequeuing: 1
After dequeue:
priority Queue content after dequeue
Data: 2, Priority: 2
Data: 3, Priority: 3
Data: 4, Priority: 4.

The dequeue function removes the element with highest priority.

ked list. The
of the
front of
lement.

to Null

CPP Circular using linked list

This program implements circular queue using linked list. The enqueue function adds elements to the rear of the queue the dequeue remove elements from the front of queue the Peek function displays the front element.

→ Initalize Queue: front and rear are initialy set to Null

→ Enqueue operations.

- Enqueue (14):
  Queue: 14
  front = rear = 14
- Enqueue (11)
  Queue: 14, 11
  front = 14, rear = 11
- Enqueue (13)
  Queue 14, 11, 13
  front = 14, rear = 13.

→ Display
  Queue: 14, 11, 13.

→ Dequeue
  After dequeue: 11, 13
  front = 11, rear = 13.

→ Display
  Queue 11, 13

→ Peek
  front = 11.

CPP Circular

→ Initalize Queue front and rear are initaly set

→ Enqueue operation

enqueue (1)

Queue : 1

front=0, rear=0

enqueue (2)

Queue : 1, 2

front=0, rear=1

enqueue(3)

Queue : 1, 2, 3

front=0, rear=2

enqueue (4)

front=0, rear=3

→ Display

Queue 1, 2, 3, 4.

→ Dequeue

Dequeue element: 1

front=1, rear=4.

→ Display

Queue 2, 3, 4

Array Size 4 the enque function adds elements to the rear queue the dequeue function removes elements from the front of queue and display function show all elements

Pop cpp

→ Initalize Stack q1 and q2 are the two queue

→ Push operations

p 1

q1: 1

p: 2

q1: 1, 2

p: 3

q1: 1, 2, 3

Size 3
Top 3

→ Pop operation
Move elements from q_1 to q_2
q_1:
q_2: 1, 2
- Swap name q_1 and q_2
q_1: 1, 2
q_2:

→ Top operation after pop:
Top: 2

→ Move elements from q_1 to q_2
q_1:
q_2: 1
- Swap the names
q_1: 1
q_2:

→ Top operation after Pop
Top 1

→ Size operation
Size 1

The Push operation is straightforward while involves moving elements between the two to maintain Stack property.

PushCpp

→ Initalize Stack: q_1 and q_2 are two queue

→ Push operation
P 1
q_1:
q_2:

- Swap q_1 and q_2
  q_1: 1
  q_2:
  P2
  q_1:
  q_2: 2, 1
- Swap q_1 and q_2
  q_1: 2, 1
  q_2:
  P3
  q_1:
  q_2: 3, 2, 1
- Swap
  q_1: 3, 2, 1
  q_2:

→ Size and Top operation
- Size 3
- Top 3.

→ Pop operation
q_1: 2, 1
q_2:

→ Top operation
- Top 2

→ Pop operation
q_1: 1
q_2:

→ Top operation
Top 1

→ Size operatio
- Size 1

The Push operation involves Pushing elements into auxiliary queue and then swapping the names of the queue. The Pop operation Pop the front element from the main q. The top operation return the front element of the main queue.