1.

s->top = -1;                // Pertanyaan 1

2.

    if (s->top == -1)           // Pertanyaan 2

3.

    if (s->top == STACKSIZE - 1) {      // Pertanyaan 3

4.

    s->top++;                           // Pertanyaan 4

5.

    s->top--;                           // Pertanyaan 5

6. Lakukan operasi-operasi sebagai berikut secara berurutan pada Stack : push(4), push(8), pop(), push(6), push(19), pop(), pop(), push(8). Apa saja nilai yang di pop dari Stack ? Bagaimana kondisi Stack (isi Stack) setelah dilakukan operasi-operasi tersebut ?

int main() {

    struct stack s;

    init(&s);

    push(&s, 4);

    push(&s, 8);

    printf("Menghapus: %d\n", pop(&s));

    push(&s, 6);

    push(&s, 19);

    printf("Menghapus: %d\n", pop(&s));

    printf("Menghapus: %d\n", pop(&s));

    push(&s, 8);

    printf("Isi stack: ");

    for (int i = 0; i <= s.top; i++) {

        printf("%d ", s.item[i]);

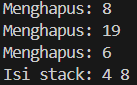
    }

    printf("\n");

    return 0;

}

Output:



6.

    if (pq->rear == MAXQUEUE - 1) {         // Pertanyaan 6

7.

    pq->rear++;                             // Pertanyaan 7

8.

    pq->front++;                            // Pertanyaan 8

9. Perhatikan fungsi empty di atas. Mengapa kondisi pengecekan untuk kondisi empty adalah pq->rear-pq->front+1==0 ? Apakah ada alternatif lainnya yang lebih baik? Jelaskan jawaban Anda.

Alternatif:

int empty(struct queue\* pq) {

    if (pq->front == 0 && pq->rear == -1)

        return TRUE;

    else

        return FALSE;

}

Pada fungsi awalnya pq->rear-pq->front+1==0 yang dimana dapat bernilai TRUE kosong meskipun keduanya tidak kosong seperti pada saat pq->rear == 3 dan pq>front == 4, sehingga lebih baik mengecek satu-satu jika kedua node masih bernilai default.

10 . Lakukan operasi-operasi sebagai berikut secara berurutan pada Queue : enqueue(4), enqueue(8), enqueue(6), enqueue(19), enqueue(8), dequeue(), dequeue(), dequeue(). Berapa nilai rear dan front-nya setelah semua operasi tersebut?. Jika dilakukan operasi ini enqueue(11), bagaimana kondisi Queue?

int main() {

    struct queue q;

    init(&q);

    insert(&q, 4);      // Rear = 0, front = 0

    insert(&q, 8);      // Rear = 1, front = 0

    insert(&q, 6);      // Rear = 2, front = 0

    insert(&q, 19);     // Rear = 3, front = 0

    insert(&q, 8);      // Rear = 4, front = 0

    remov(&q);          // Rear = 4, front = 1

    remov(&q);          // Rear = 4, front = 2

    remov(&q);          // Rear = 4, front = 3

    remov(&q);          // Rear = 4, front = 4

    printf("Setelah operasi:\nFront: %d, Rear: %d\n", q.front, q.rear);

    insert(&q, 11);     // Rear = 5, front = 4  (Queue Overflows)

    return 0;

}

Output:

