

sesatmaxxing

struktur data

the one and only

kafeyangasli

12/10/25

1. Perhatikan tabel berikut:

No.	Pemesan	Menu dan Jumlah	TotalPesanan	TotalHarga	Pembayaran
1.	Andi	Bakso (3), Pizza (1)	3	195000	QRIS
2.	Tono	EsTeh (4), Soto (2)	6	25000	Tunai
3.	Kirby	DimsumMentai (15)	15	1900000	Debit
...	...	...	...	...	...

- a. Buatlah definisi type untuk menyimpan data pesanan sesuai tabel di atas!

```

DEKLARASI TIPE
type Menu: <Nama: string, Jumlah: integer>
type Pesanan: <Pemesan: string, MDJ: array[1..10] of Menu,
               TotalPesanan: integer ≥ 0, TotalHarga: integer ≥ 0,
               Pembayaran: string>
type Tabel: <size: integer, wadah: array[1..10] of Pesanan>

```

- b. Buat deklarasi, kamus lokal, dan algoritma untuk prosedur tambah pesanan ke tabel jika pelanggan belum pernah memesan.

```

procedure AddPesanan(input/output: T: Tabel, input: P: Pesanan)
{I.S: T dan P terdefinisi}
{F.S: P ditambah pada tabel T dengan indeks T.size + 1 jika belum
penuh dan P.Pemesan tidak ada di tabel}

Kamus Lokal
  i: integer
  found: boolean

Algoritma
  if not isFullTabel(T) then
    found ← false
    i ← 1
    while (not found and i ≤ T.size) do
      found ← T.wadah[i].Pemesan = P.Pemesan
      {Jika sama, maka akan False yang berarti
      Pemesan atas nama P.Pemesan pernah memesan}
      i ← i + 1
    {endWhile}
    if not found then {P.Pemesan belum pernah memesan}
      T.size ← T.size + 1
      T.wadah[T.size] ← P
    {endIf}
  {endIf}

```

2. Prosedur memasukkan P ke Stack QRIS, Tunai, atau Debit.

```
procedure bagiPembayaran(input/output: QRIS, Tunai, Debit: Stack,  
                        input: P: Pesanan)  
{I.S: QRIS, Tunai, Debit, dan P terdefinisi  
{F.S: Salah satu dari ketiga Stack dipush dengan P atau tidak  
semuanya.}  
Kamus Lokal  
-  
Algoritma  
  depend on P.Pembayaran  
    P.Pembayaran = "QRIS": push(QRIS, P)  
    P.Pembayaran = "Tunai": push(Tunai, P)  
    P.Pembayaran = "Debit": push(Debit, P)
```

3. Mengantrekan pesanan berdasarkan jumlah pesanan

```
procedure bagiPesanan(input/output: Q: Queue, input: P: Pesanan)  
{I.S: Q terdefinisi}  
{F.S: P diqueue ke Q berdasarkan jumlah pesannya (terbanyak di dekat  
Head)}  
  
Kamus Lokal  
  i: integer  
  
Algoritma  
  if not isFullQueue(Q) then  
    if isEmptyQueue(Q) then  
      Q.head  $\leftarrow$  1  
      Q.tail  $\leftarrow$  1  
      Q.wadah[1]  $\leftarrow$  P  
    else  
      i  $\leftarrow$  Q.tail  
      while (P.JumlahPesanan > Q.wadah[i].JumlahPesanan and i > 0)  
do  
        Q.wadah[i + 1]  $\leftarrow$  Q.wadah[i]  
        i  $\leftarrow$  i - 1  
      {endWhile}  
      Q.wadah[i + 1]  $\leftarrow$  P  
      Q.tail  $\leftarrow$  Q.tail + 1  
    {endIf}  
  {endIf}
```

Idenya adalah selama elemen pada Tail memiliki JumlahPesanan yang lebih kecil, maka dia akan dipindahkan ke Tail + 1, lalu akan lanjut ke elemen Tail - 1 sampai elemen Pesanan memiliki JumlahPesanan yang sama dengan atau lebih besar dari P.

Ketika sudah berhenti, maka dia akan menempati tempat yang kosong tersebut. Untuk JumlahPesanan yang sama, karena dia berhenti pada saat JumlahPesanan sudah ekivalen, maka sudah pasti urutannya berdasarkan urutan datangnya Pesanan.