Latihan Modul 9

STRUKTUR DATA - Ganjil 2023/2024

"Queue"

Sebuah pusat kesehatan bernama **"Klinik Sehat Yogi"** sedang mengadakan **program vaksinasi massal** selama satu bulan. Program ini menarik ratusan warga dari berbagai kategori yang mendaftar setiap harinya. Klinik menerapkan **sistem antrean berbasis *queue* yang kompleks** untuk memastikan pengelolaan alur dan prioritas warga secara efektif.

Aturan pendaftaran

1. Setiap hari, warga yang berhasil mendaftar akan menerima **kode antrean unik** yang valid untuk satu hari. Setiap hari ada **tiga queue terpisah** yang disediakan:

    a. **Queue Prioritas (Prioritas Tinggi)**: Dikhususkan untuk **lansia (usia ≥ 60 tahun)**, **tenaga kesehatan**, dan **ibu hamil**. Mereka ditempatkan di urutan pertama untuk mendapatkan layanan.
    b. **Queue Normal (Prioritas Rendah)**: Untuk warga yang tidak memenuhi syarat prioritas.
    c. **Queue Tunggu**: Untuk warga yang tidak bisa dilayani pada hari tersebut karena **kapasitas klinik penuh**. Warga di antrean tunggu akan secara otomatis dipindahkan ke antrean prioritas atau antrean normal pada hari berikutnya berdasarkan prioritas mereka.

1. **Kapasitas Klinik Sehat Yogi adalah 100 orang per hari. Jika jumlah warga yang hadir melebihi kapasitas, warga di Queue Tunggu akan mendapatkan prioritas pelayanan pada hari berikutnya.**

2. **Batas Waktu Antrean**

    Jika seorang warga menunggu di antrean lebih dari **2 jam**, mereka akan mendapatkan prioritas tambahan dan dipindahkan ke **Queue Prioritas**. Sistem ini menandai antrean yang telah lama menunggu dan secara otomatis mengatur ulang prioritas setiap jam.

3. **Prioritas Dinamis**

    Jika seorang warga dengan kondisi darurat medis datang ke klinik (misalnya, mengalami reaksi alergi atau tekanan darah tinggi saat berada di antrean), warga tersebut otomatis dipindahkan ke posisi paling depan di **Queue Prioritas** untuk segera

dilayani. Warga lain dalam antrean prioritas akan tetap berada di antrean sesuai urutan awal mereka.

Buatlah ADT stack pada file “queue_vaksinasi.h” dengan struktur berikut:

Notasi Algoritmik	Bahasa C++

type Queue :
< head, tail : Address >
 	
type ElemQ :
< info : Infotype,
next : Address >
 	
type Infotype :
< nama: string,
usia: integer,
pekerjaan: string,
prioritas: string, // "Prioritas Tinggi", "Normal", atau "Tunggu"
nomor_antrean: integer,
waktu_daftar: integer, // waktu dalam menit
kondisi_darurat: boolean >
	struct Queue {
	ElemQ *head;
	ElemQ *tail;
	};
	
	struct ElemQ {
	Infotype info;
	ElemQ *next;
	};
	
	struct Infotype {
	string nama;
	int usia;
	string pekerjaan;
	bool prioritas;
	int nomor_antrean;
	bool kondisi_darurat;
	};

Prosedur dan fungsi yang digunakan adalah sebagai berikut

Primitive Procedures/Functions	
Notasi Algoritmik	Bahasa C++

procedure createQueue(in/out Q : Queue)
{ I.S. -
F.S. terbentuk queue dengan head dan tail NIL }
	void createQueue(Queue &Q) {
	head(Q) = nill;
	tail(Q) = nill;
	}

function isEmpty(Q : Queue) -> boolean
{ Mengembalikan nilai true jika queue kosong }
	bool isEmpty(Queue Q) {
	return head(Q) == nill;
	}

function createElemQueue(nama: string, usia: integer, pekerjaan: string) -> Address
{ Mengembalikan sebuah pointer yang menunjuk ke elemen queue (ElemQ) }
	ElemQ* createElemQueue(string nama, int usia, string pekerjaan, int nomor_antrean) {
	ElemQ *P = new ElemQ;
	info(P).nama = nama;
	info(P).usia = usia;
	info(P).pekerjaan = pekerjaan;

	info(P).prioritas = (usia >= 60 || pekerjaan == "tenaga kesehatan");
	info(P).nomor_antrean = nomor_antrean;
	info(P).kondisi_darurat = false;
	next(P) = nill;
	return P;
	}

procedure enqueue(in/out Q : Queue, in P : Address)
{ *I.S.*: Queue Q mungkin kosong, pointer P menunjuk ke elemen queue yang akan ditambahkan
 	
F.S.: Elemen yang ditunjuk oleh P ditambahkan ke dalam queue Q sesuai aturan:
 	
• Jika info.prioritas = true, elemen dimasukkan di depan antrean setelah elemen prioritas lainnya
• Jika info.prioritas = false, elemen dimasukkan di belakang antrean
}
	void enqueue(Queue &Q, ElemQ *P) {
	if (isEmpty(Q)) {
	head(Q) = P;
	tail(Q) = P;
	} else if (info(P).prioritas) {
	if (!info(head(Q)).prioritas) {
	next(P) = head(Q);
	head(Q) = P;
	} else {
	ElemQ *temp = head(Q);
	while (next(temp) != nill && info(next(temp)).prioritas) {
	temp = next(temp);
	}
	next(P) = next(temp);
	next(temp) = P;
	if (next(P) == nill) {
	tail(Q) = P;
	}
	}
	} else {
	next(tail(Q)) = P;
	tail(Q) = P;
	}
	}

procedure dequeue(in/out Q : Queue, in/out P : Address)
 	
{ *I.S.*: Queue Q mungkin kosong
 	
F.S.: Jika Q tidak kosong, elemen pertama dihapus dari Q dan P menunjuk ke elemen tersebut. Jika Q kosong, P bernilai NIL dan mencetak "Semua warga telah terlayani." }
	void dequeue(Queue &Q, ElemQ *&P) {
	if (isEmpty(Q)) {
	P = nill;
	cout << "Semua warga telah terlayani." << endl;
	} else {
	P = head(Q);
	head(Q) = next(head(Q));
	if (head(Q) == nill) {
	tail(Q) = nill;
	}
	next(P) = nill;

	} }

function front(in Q : Queue) -> Address
{ Mengembalikan pointer ke elemen pertama dalam queue Q }
	ElemQ* front(Queue Q) {
	return head(Q);
	}

function back(in Q : Queue) -> Address
{ Mengembalikan pointer ke elemen terakhir dalam queue Q }
	ElemQ* back(Queue Q) {
	return tail(Q);
	}

function size(in Q : Queue) -> integer
{ Mengembalikan jumlah elemen dalam queue Q dengan menghitung dari head hingga tail }
	int size(Queue Q) {
	int count = 0;
	ElemQ *temp = head(Q);
	while (temp != nill) {
	count++;
	temp = next(temp);
	}
	return count;
	}

Buat prosedur/fungsi non primitif sebagai berikut. Perhatikan bahwa implementasi dari prosedur/fungsi non primitif hanya boleh menggunakan prosedur/fungsi primitive saja.

procedure printInfo(in Q : Queue)
{ I.S. terdefinisi queue mungkin kosong
F.S. Menampilkan seluruh informasi warga yang ada dalam antrean Q dari head hingga tail tanpa mengubah isi atau urutan dalam antrean.}

procedure serveQueue(in/out Q : Queue)

{I.S.: Queue Q mungkin kosong atau terisi dengan data warga.

F.S.: Melayani warga dalam antrean Q hingga kapasitas maksimal per hari (100 orang) atau hingga antrean kosong.

- Setiap elemen yang dilayani dihapus dari antrean.
- Menampilkan pesan konfirmasi vaksinasi untuk setiap warga yang dilayani.
- Jika kapasitas harian tercapai (100 orang), menampilkan pesan bahwa kapasitas telah penuh.
- Jika masih ada warga di antrean, menampilkan pesan bahwa warga yang belum terlayani diminta kembali besok.

}

procedure reassignQueue(in/out Q : Queue)

{I.S.: Queue Q terdefinisi, mungkin terdapat warga yang belum terlayani dari antrean sebelumnya.

F.S.: Memindahkan warga dari antrean tunggu ke antrean prioritas atau antrean normal pada hari berikutnya sesuai kategori prioritas masing-masing warga.

- Warga prioritas ditempatkan di depan antrean prioritas.
- Warga non-prioritas ditempatkan di belakang antrean normal.

}

procedure checkWaitingTime(in/out Q : Queue, waktu_sekarang : integer)

{I.S.: Queue Q terdefinisi, mungkin terdapat warga yang menunggu terlalu lama dalam antrean.

F.S.: Mengecek waktu tunggu setiap warga dalam antrean Q.

- Jika waktu tunggu warga lebih dari 2 jam (120 menit), mereka dipindahkan ke Queue Prioritas jika belum berada di sana.
- Mengatur ulang posisi warga dalam antrean sesuai dengan perubahan prioritas akibat waktu tunggu.

}

procedure emergencyHandle(in/out Q : Queue, nomor_antrean : integer)

{I.S.: Queue Q terdefinisi dan berisi warga dalam antrean.

F.S.: Jika ditemukan warga dengan nomor_antrean yang sesuai dan mengalami kondisi darurat, maka:

- kondisi_darurat warga tersebut diubah menjadi true.
- Warga tersebut dipindahkan ke posisi paling depan di Queue Prioritas untuk dilayani segera.
- Jika nomor_antrean tidak ditemukan, menampilkan pesan bahwa warga dengan nomor tersebut tidak ada dalam antrean.

}

procedure updatePriority(in/out Q : Queue)

{I.S.: Queue Q terdefinisi dan berisi warga dalam antrean.

F.S.: Mengatur ulang antrean setiap jam untuk memastikan warga yang memenuhi syarat prioritas mendapatkan tempat sesuai dengan tingkat urgensi.

- Warga yang memiliki prioritas atau menunggu lebih dari 2 jam dipindahkan ke depan antrean.
- Warga dalam kondisi darurat berada di paling depan antrean.
- Warga non-prioritas berada di belakang antrean setelah warga prioritas.

}

function findAndRemove(in/out Q : Queue, nomor_antrean : integer) -> Address

{ I.S.: Queue Q terdefinisi dan berisi warga dalam antrean.

F.S.: Mengembalikan alamat elemen yang berisi nomor_antrean yang sesuai dan menghapus elemen tersebut dari antrean Q.

- Jika elemen ditemukan, elemen tersebut dihapus dari antrean dan alamatnya dikembalikan.
- Jika elemen tidak ditemukan, mengembalikan NIL dan menampilkan pesan bahwa warga dengan nomor antrean tersebut tidak ada dalam antrean.

}

Buat program utama (main.cpp). Program ini membuat representasi antrean vaksinasi di Klinik Sehat Yogi dan menjalankan prosedur serveQueue (pelayanan vaksinasi) hingga antrean kosong atau mencapai kapasitas harian maksimal.

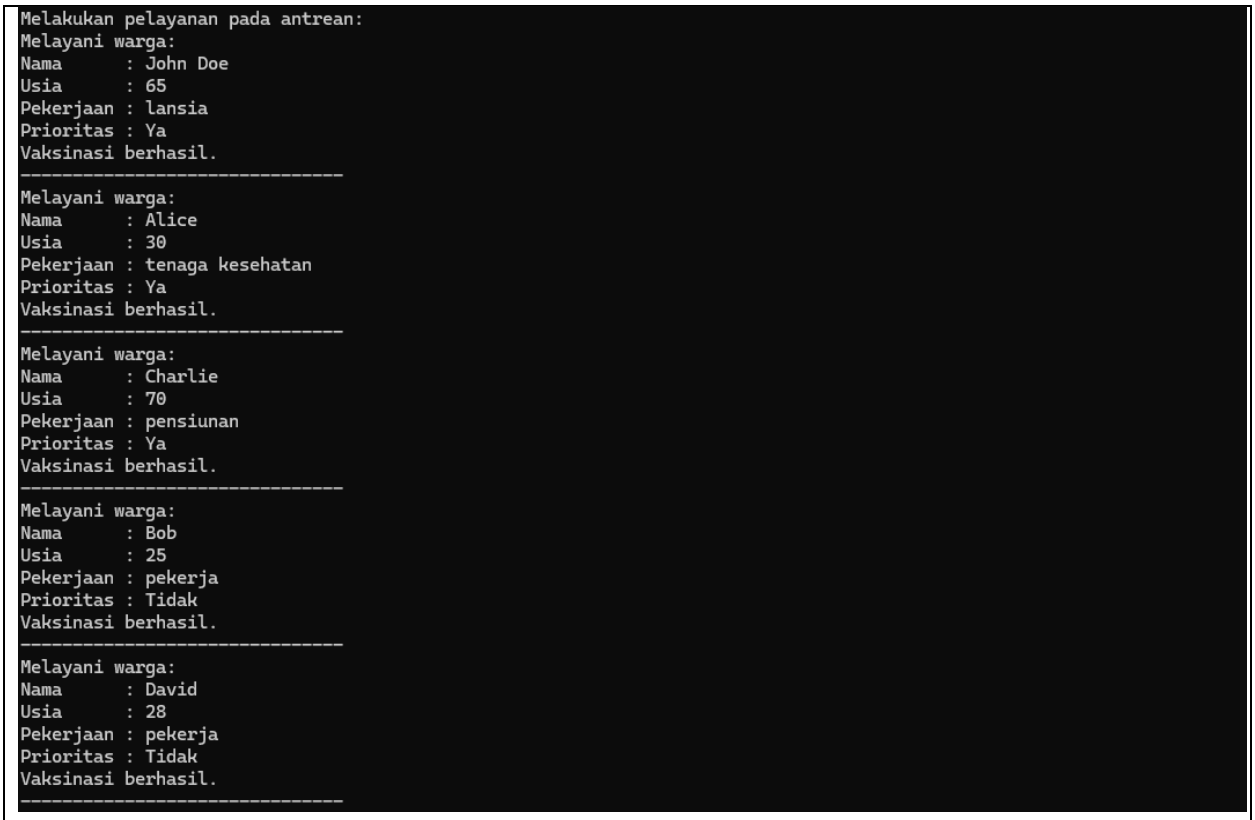
Main.cpp

```
int main() {
// Membuat antrean
Queue Q;
createQueue(Q);

// Menambahkan beberapa elemen ke dalam queue
ElemQ* P1 = createElemQueue("John Doe", 65, "lansia", 1);
ElemQ* P2 = createElemQueue("Alice", 30, "tenaga kesehatan", 2);
ElemQ* P3 = createElemQueue("Bob", 25, "pekerja", 3);
ElemQ* P4 = createElemQueue("Charlie", 70, "pensiunan", 4);
ElemQ* P5 = createElemQueue("David", 28, "pekerja", 5);

enqueue(Q, P1);
```

```
enqueue(Q, P2);
enqueue(Q, P3);
enqueue(Q, P4);
enqueue(Q, P5);

// Menampilkan isi queue
cout << "Isi antrean awal:" << endl;
printInfo(Q);
```

```
Isi antrean awal:
Daftar Antrean:
Nama: John Doe
Usia: 65
Pekerjaan: lansia
Prioritas: Ya
Nomor Antrean: 1
-------------------------
Nama: Alice
Usia: 30
Pekerjaan: tenaga kesehatan
Prioritas: Ya
Nomor Antrean: 2
-------------------------
Nama: Charlie
Usia: 70
Pekerjaan: pensiunan
Prioritas: Ya
Nomor Antrean: 4
-------------------------
Nama: Bob
Usia: 25
Pekerjaan: pekerja
Prioritas: Tidak
Nomor Antrean: 3
-------------------------
Nama: David
Usia: 28
Pekerjaan: pekerja
Prioritas: Tidak
Nomor Antrean: 5
-------------------------
```

```
// Melayani antrean
cout << "\nMelakukan pelayanan pada antrean:" << endl;
serveQueue(Q);
```




```
// Memeriksa antrean setelah pelayanan
cout << "\nIsi antrean setelah pelayanan:" << endl;
printInfo(Q);

// Menambahkan elemen baru untuk simulasi pengaturan ulang prioritas

ElemQ* P6 = createElemQueue("Edward", 22, "pekerja", 6);
enqueue(Q, P6);

// Simulasi reassignQueue untuk mengatur ulang antrean ke prioritas
cout << "\nMengatur ulang antrean berdasarkan prioritas:" << endl;
reassignQueue(Q);
printInfo(Q);

// Simulasi kondisi warga yang menunggu lebih dari 2 jam
cout << "\nMemeriksa waktu tunggu dan mengubah prioritas jika lebih dari 2 jam:" << endl;
checkWaitingTime(Q, 130); // Asumsikan waktu sekarang 130 menit dari nomor antrean
pertama

printInfo(Q);
```

```
// Menangani kondisi darurat untuk seorang warga cout << "\nMenangani kondisi darurat
untuk warga dengan nomor antrean 5:" << endl;
emergencyHandle(Q, 5);
printInfo(Q);

// Update prioritas warga setiap jam
cout << "\nMengupdate prioritas antrean setiap jam:" << endl;
updatePriority(Q);
printInfo(Q);
```

```
Isi antrean setelah pelayanan:
Antrean kosong.

Mengatur ulang antrean berdasarkan prioritas:
Daftar Antrean:
Nama: Edward
Usia: 22
Pekerjaan: pekerja
Prioritas: Tidak
Nomor Antrean: 6
-------------------------

Memeriksa waktu tunggu dan mengubah prioritas jika lebih dari 2 jam:
Daftar Antrean:
Nama: Edward
Usia: 22
Pekerjaan: pekerja
Prioritas: Ya
Nomor Antrean: 6
-------------------------

Menangani kondisi darurat untuk warga dengan nomor antrean 5:
Warga dengan nomor antrean 5 tidak ditemukan.
Daftar Antrean:
Nama: Edward
Usia: 22
Pekerjaan: pekerja
Prioritas: Ya
Nomor Antrean: 6
-------------------------

Mengupdate prioritas antrean setiap jam:
Daftar Antrean:
Nama: Edward
Usia: 22
Pekerjaan: pekerja
Prioritas: Ya
Nomor Antrean: 6
-------------------------
```

```
// Mencari dan menghapus warga dengan nomor antrean tertentu
cout << "\nMenghapus warga dengan nomor antrean 3:" << endl;
ElemQ* removedElem = findAndRemove(Q, 3);

if (removedElem) {
cout << "Warga yang dihapus: " << info(removedElem).nama << endl;
}

printInfo(Q);
```

// Mengecek ukuran queue cout << "\nUkuran antrean saat ini: " << size(Q) << endl;

```
Menghapus warga dengan nomor antrean 3:
Warga dengan nomor antrean 3 tidak ditemukan dalam antrean.
Daftar Antrean:
Nama: Edward
Usia: 22
Pekerjaan: pekerja
Prioritas: Ya
Nomor Antrean: 6
------------------------

Ukuran antrean saat ini: 1
```

return 0; }