LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
TUGAS OTS QUEUE

Di susun oleh :

CAHYANING ERDINIRA WIDIYA LESTARI

1203220089

IF 03-02

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY SURABAYA

2024

1. **Source code beserta penjelasannya**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 100

//struktur untuk node dalam queue
typedef struct {
    int data;
} Node;

//struktur untuk queue
typedef struct {
    Node *array[MAX_SIZE];
    int front, rear;
} Queue;

//fungsi untuk membuat queue baru
Queue *createQueue() {
    Queue *queue = (Queue *)malloc(sizeof(Queue));
    queue->front = -1;
    queue->rear = -1;
    return queue;
}

//fungsi untuk menambahkan elemen ke dalam queue
void enqueue(Queue *queue, Node *item) {
    if (queue->rear == MAX_SIZE - 1) {
        printf("Antrian penuh\n");
        return;
    }
    if (queue->front == -1)
        queue->front = 0;
    queue->rear++;
    queue->array[queue->rear] = item;
}

//fungsi untuk menghapus elemen dari queue
Node *dequeue(Queue *queue) {
    if (queue->front == -1 || queue->front > queue->rear) {
        printf("Antrian kosong\n");
        return NULL;
    }
    Node *item = queue->array[queue->front];
    queue->front++;
    return item;
```

```c
}

//fungsi untuk memeriksa apakah queue kosong
int isEmpty(Queue *queue) {
    return queue->front == -1 || queue->front > queue->rear;
}

//fungsi untuk menangani pasien sesuai dengan tingkat keparahan
void handlePatients(int severity[], int n) {
    Queue *generalDoctorQueue = createQueue();
    Queue *specialistDoctorQueue = createQueue();

    for (int i = 0; i < n; i++) {
        Node *patient = (Node *)malloc(sizeof(Node));
        patient->data = severity[i];
        if (severity[i] <= 5) {
            enqueue(generalDoctorQueue, patient);
            printf("ditangani dokter umum\n");
        } else {
            enqueue(specialistDoctorQueue, patient);
            printf("ditangani dokter spesialis\n");
        }
    }

    //membersihkan queue setelah penanganan pasien
    while (!isEmpty(generalDoctorQueue)) {
        free(dequeue(generalDoctorQueue));
    }
    while (!isEmpty(specialistDoctorQueue)) {
        free(dequeue(specialistDoctorQueue));
    }

    //menghapus queue setelah digunakan
    free(generalDoctorQueue);
    free(specialistDoctorQueue);
}

int main() {
    char input[MAX_SIZE];
    int severity[MAX_SIZE];
    int n = 0;

    printf("Masukkan antrian pasien: ");
    fgets(input, sizeof(input), stdin);

    //memecah string input menjadi angka-angka dan menyimpannya dalam array
severity
    char *token = strtok(input, " ");
```

```
    while (token != NULL) {
        severity[n++] = atoi(token);
        token = strtok(NULL, " ");
    }

    handlePatients(severity, n);
    return 0;
}
```

2. **Ouput program**

    - **Input simple 1**

    ```
    Masukkan antrian pasien: 5 1 4 8 9 7 1 2 9 4
    ```

    - **Output simple 1**

    ```
    ditangani dokter umum
    ditangani dokter umum
    ditangani dokter umum
    ditangani dokter spesialis
    ditangani dokter spesialis
    ditangani dokter spesialis
    ditangani dokter umum
    ditangani dokter umum
    ditangani dokter spesialis
    ditangani dokter umum
    ```