Nama: Najwa Kurnia

Nim: 23343058

Matkul: Praktikum Struktur Data

PROGRAM ALGORITMA BREADTH FIRST SEARCH

```
struct Graph{
    int numNode;
    int matriksKedekatan[maxNode][maxNode];
};
struct Queue{
    int items[maxNode];
    int depan;
    int belakang;
};
```

Definisi dari struktur graph yang berisi jumlah node dan matriks untuk mempresentasikan hubungan antara node, dan struktur queue yang mempresentasikan struktur data antrian.

```
struct Queue* createQueue(){
    struct Queue* queue=(struct Queue*)malloc(sizeof(struct Queue));
    queue->depan= -1;
    queue->belakang= -1;
    return queue;
}
```

Fungsi untuk membuat antrian secara dinamis di heap dan menginisialisasi penunjuk depan dan belakangnya.

```
bool isEmpty(struct Queue* queue){
    if(queue->belakang== -1){
        return true;
    }else{
        return false;
    }
}
```

Fungsi untuk memeriksa apakah antrian kosong.

```
void enqueue(struct Queue* queue, int value){
    if(queue->belakang==maxNode -1){
        printf("\nAntrian penuh!!");
    }else{
        if(queue->depan== -1){
            queue->depan=0;
        }
```

```
        queue->belakang++;
        queue->items[queue->belakang]=value;
    }
}
```

Fungsi untuk menambahkan sebuah elemen ke antrian dengan memeriksa terlebih dahulu apakah antrian penuh atau tidak, kalua tidak maka elemen dapat ditambahkan.

```
int dequeue(struct Queue* queue){
    int item;
    if(isEmpty(queue)){
        printf("Antrian kosong\n");
        item= -1;
    }else{
        item=queue->items[queue->depan];
        queue->depan++;
        if(queue->depan > queue->belakang){
            queue->depan=queue->belakang= -1;
        }
    }
    return item;
}
```

Fungsi ini untuk menghapus sebuah elemen dari antrian, dengan membuat sebuah variable baru untuk menampung antrian yang akan dihapus, lalu memindahkan antrian tersebut ke elemen selanjutnya.

```
void initGraph(struct Graph* graph, int node){
    graph->numNode=node;
    for(int i=0; i<node; i++){
        for(int j=0; j<node; j++){
            graph->matriksKedekatan[i][j]=0;
        }
    }
}
```

Fungsi untuk menginisialisasi graf dengan jumlah node. Dalam graf yang direpresentasikan menggunakan matriks, inisialisasi ini dilakukan dengan mengatur semua elemen matriks menjadi 0, yang menandakan bahwa pada awalnya tidak ada sambungan antara node-node tersebut.

```
void tambahEdge(struct Graph* graph, int src, int dest){
    graph->matriksKedekatan[src][dest]=1;
    graph->matriksKedekatan[dest][src]=1;
}
```

Fungsi ini untuk menambahkan edge antara dua node dalam graf yang direpresentasikan menggunakan matriks. Nilai 1 pada matriks menandakan adanya sambungan antara dua node, sedangkan nilai 0 menandakan tidak adanya sambungan.

```
void BFS(struct Graph* graph, int start){
    bool visit[maxNode]={false};
    struct Queue* queue= createQueue();
    visit[start]=true;
    enqueue(queue, start);
    while(!isEmpty(queue)){
        int currentNode= dequeue(queue);
        printf("%d ", currentNode);
        for(int i=0; i<graph->numNode; i++){
            if(graph->matriksKedekatan[currentNode][i]==1 && !visit[i]){
                visit[i]=true;
                enqueue(queue, i);
            }
        }
    }
}
```

Fungsi ini untuk melakukan penelusuran graf menggunakan algoritma Breath First Search (BFS). Algoritma ini mengunjungi semua node dalam graf secara terstruktur, dimulai dari node awal yang ditentukan.

```
int main(){
    struct Graph graph;
    int node, edge, i, start;
    printf("Masukkan jumlah node dalam grafik: ");
    scanf("%d", &node);
    initGraph(&graph, node);

    printf("Masukkan jumlah sisi dalam grafik: ");
    scanf("%d", &edge);

    for(i=0; i<edge; i++){
        int src, dest;
        printf("Masukkan edge %d tujuan sumber: ", i+1);
        scanf("%d %d", &src, &dest);
        tambahEdge(&graph, src, dest);
    }
    printf("Masukkan node awal untuk traversal BFS: ");
    scanf("%d", &start);
    printf("Traversal BFS: ");
    BFS(&graph, start);
    return 0;
}
```

Fungsi utama ini untuk memanggil semua fungsi yang telah didefinisikan, serta untuk meminta input dari pengguna, membangun graf berdasarkan input tersebut dan menjalankan penelusuran BFS dari node yang ditentukan oleh pengguna.