Nama : Gede Satyamahinsa Prastita Uttama

NIM : 1203220054

Kelas : IF-02-03

TUGAS GRAPH

1. Source Code

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

// Deklarasi graph dengan menggunakan struct yang memiliki alias Graph.
// Di dalamnya, terdapat variabel vertex bertipe data integer untuk menyimpan nilai,
// variabel jumlah_tetangga bertipe data integer untuk menyimpan banyak tetangga,
// variabel pointer tetangga untuk menyimpan tetangga yang berhubungan.
typedef struct {
  int vertex;
  int jumlah_tetangga;
  int *tetangga;
} Graph;

// buatGraph() => digunakan untuk membuat sebuah graph dengan banyak vertex yang telah ditentukan.
Graph* buatGraph(int banyak_vertex) {
  Graph* graph = (Graph*)malloc(banyak_vertex * sizeof(Graph));

  // Inisialisasi setiap vertex dengan nilai dari 0 hingga banyak vertex - 1.
  for (int i = 0; i < banyak_vertex; i++) {
    graph[i].vertex = i;
    graph[i].jumlah_tetangga = 0;
    graph[i].tetangga = NULL;
  }

  // Mengembalikan graph yang telah dibuat.
  return graph;
}

// tambahEdge() => digunakan untuk menambahkan edge antara dua vertex, yaitu asal (src) dan tujuan (dest).
void tambahEdge(Graph* graph, int src, int dest) {
  // Mengalokasikan ulang memori pada tetangga src sesuai dengan jumlah tetangga yang sudah diupdate.
  graph[src].tetangga = (int*)realloc(graph[src].tetangga, (graph[src].jumlah_tetangga + 1) * sizeof(int));
  // Menambahkan nilai dest ke dalam tetangga dari src
  graph[src].tetangga[graph[src].jumlah_tetangga] = dest;
  // Menambahkan jumlah_tetangga dengan 1.
  graph[src].jumlah_tetangga++;

  // Mengalokasikan ulang memori pada tetangga dest sesuai dengan jumlah tetangga yang sudah diupdate.
  graph[dest].tetangga = (int*)realloc(graph[dest].tetangga, (graph[dest].jumlah_tetangga + 1) * sizeof(int));
  // Menambahkan nilai src ke dalam tetangga dest
  graph[dest].tetangga[graph[dest].jumlah_tetangga] = src;
  // Menambahkan jumlah_tetangga dengan 1.
  graph[dest].jumlah_tetangga++;
}
```

```c
int main() {
  // Deklarasi variabel jumlah_orang, banyak_tetangga, src, dan dest bertipe data integer.
  int jumlah_orang, banyak_tetangga, src, dest;

  // Meminta user untuk memasukkan jumlah orang dan banyak tetangga.
  scanf("%d %d", &jumlah_orang, &banyak_tetangga);

  // Memanggil fungis buatGraph() untuk membuat sebuah graph dengan banyak vertex sesuai jumlah orang.
  Graph* graph = buatGraph(jumlah_orang);

  // Melakukan perulangan hingga user mengetikkan nilai -1 pada src dan dest.
  while(1) {
    // Meminta user untuk memasukkan edge setiap vertex melalui input.
    scanf("%d %d", &src, &dest);

    // Perulangan akan berhenti ketika src dan dest bernilai -1.
    if(src == -1 && dest == -1) {
      break;
    }

    // Memanggil fungsi tambahEdge() untuk menambahkan edge di antara vertex src dan dest.
    tambahEdge(graph, src, dest);
  }

  // Perulangan dilakukan untuk memeriksa apakah seseorang termasuk wibu nolep atau bukan.
  for(int i = 0; i < jumlah_orang; i++) {
    // Jika ada seseorang (vertex) memiliki banyak tetangga (edge) yang lebih dari
    // aturan yang telah ditetapkan, maka seseorang (vertex) tersebut merupakan wibu nolep.
    if(graph[i].jumlah_tetangga < banyak_tetangga) {
      printf("Vertex %d adalah wibu nolep", i);
    }
  }

  // Menghapus atau membebaskan memori graph secara keseluruhan.
  for (int i = 0; i < jumlah_orang; i++) {
    free(graph[i].tetangga);
  }
  free(graph);

  return 0;
}
```

Output

```
5 2
0 1
0 4
0 3
1 2
1 3
2 3
-1 -1
Vertex 4 adalah wibu nolep
```

2. Source Code

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

// Deklarasi graph dengan menggunakan struct yang memiliki alias Graph.
// Di dalamnya, terdapat variabel vertex bertipe data integer untuk menyimpan nilai,
// variabel jumlah_tetangga bertipe data integer untuk menyimpan banyak tetangga,
// variabel pointer tetangga untuk menyimpan tetangga yang berhubungan.
typedef struct {
  int vertex;
  int simpang;
  int *persimpangan;
} Graph;

// buatGraph() => digunakan untuk membuat sebuah graph dengan banyak vertex yang telah ditentukan.
Graph* buatGraph(int banyak_vertex) {
  Graph* graph = (Graph*)malloc(banyak_vertex * sizeof(Graph));

  // Inisialisasi setiap vertex dengan nilai dari 0 hingga banyak vertex - 1.
  for (int i = 0; i < banyak_vertex; i++) {
    graph[i].vertex = i;
    graph[i].simpang = 0;
    graph[i].persimpangan = NULL;
  }

  // Mengembalikan graph yang telah dibuat.
  return graph;
}

// tambahEdge() => digunakan untuk menambahkan edge antara dua vertex, yaitu asal (src) dan tujuan (dest).
void tambahEdge(Graph* graph, int src, int dest) {
  // Mengalokasikan ulang memori pada persimpangan src sesuai dengan jumlah persimpangan yang sudah diupdate.
  graph[src].persimpangan = (int*)realloc(graph[src].persimpangan, (graph[src].simpang + 1) * sizeof(int));
  // Menambahkan nilai dest ke dalam persimpangan dari src
  graph[src].persimpangan[graph[src].simpang] = dest;
  // Menambahkan simpang dengan 1.
  graph[src].simpang++;

  // Mengalokasikan ulang memori pada persimpangan dest sesuai dengan jumlah persimpangan yang sudah diupdate.
  graph[dest].persimpangan = (int*)realloc(graph[dest].persimpangan, (graph[dest].simpang + 1) * sizeof(int));
  // Menambahkan nilai src ke dalam persimpangan dest
  graph[dest].persimpangan[graph[dest].simpang] = src;
  // Menambahkan simpang dengan 1.
  graph[dest].simpang++;
}
```

```c
int main() {
  // Deklarasi variabel banyak_persimpangan, umur, src, dan dest bertipe data integer.
  int banyak_persimpangan, umur, src, dest;

  // Meminta user untuk memasukkan jumlah persimpangan dan umur Zoro saat ini
  scanf("%d", &banyak_persimpangan);
  scanf("%d", &umur);

  // Memanggil fungis buatGraph() untuk membuat sebuah graph dengan banyak vertex sesuai jumlah orang.
  Graph* graph = buatGraph(banyak_persimpangan);

  // Melakukan perulangan hingga user mengetikkan nilai -1 pada src dan dest.
  while (1) {
    scanf("%d %d", &src, &dest);

    // Perulangan akan berhenti ketika src dan dest bernilai -1.
    if (src == -1 && dest == -1) {
      break;
    }

    // Memanggil fungsi tambahEdge() untuk menambahkan edge di antara vertex src dan dest.
    tambahEdge(graph, src, dest);
  }

  printf("Persimpangan yang harus dihindari oleh Roronoa Zoro adalah:\n");
  // Menentukan persimpangan yang harus dihindari berdasarkan umur Roronoa Zoro.
  for(int i = 0; i < banyak_persimpangan; i++) {
    if (graph[i].simpang > umur) {
      printf("Persimpangan %d\n", i);
    }
  }

  // Menghapus atau membebaskan memori graph secara keseluruhan.
  for (int i = 0; i < banyak_persimpangan; i++) {
    free(graph[i].persimpangan);
  }
  free(graph);

  return 0;
}
```

Output

```
5
2
0 1
0 4
1 2
1 3
1 4
2 3
3 4
-1 -1
Persimpangan yang harus dihindari oleh Roronoa Zoro adalah:
Persimpangan 1
Persimpangan 3
Persimpangan 4
```