T4. Practice - OOP Graph

1. Gambarkan peta sekitar rumah kalian dengan minimal 10 titik dalam bentuk graf berarah (20 poin)

2. Dengan menggunakan Class Peta yang telah saya contohkan, implementasikan peta yang telah kalian buat (nomor 1) ke dalam sebuah program dengan representasi graf adjacency list (20 poin)

```cpp
#include <iostream>
#include <list>
#include <stack>
#include <stdio.h>
using namespace std;

class Peta
{

private:
  // Property
  int jumlah_titik;
  list<int> *adjacency_list;
  int **adjacency_matrix;

public:
  // Constructor
  Peta(int jumlah_titik)
  {
    this->jumlah_titik = jumlah_titik;
    this->inisialisasiAdjList(jumlah_titik);
```

```
    this->inisialisasiAdjMatrix(jumlah_titik);
  }

  // Destructor
  ~Peta()
  {
    delete[] adjacency_list;
  }

  // Fungsi untuk inisialisasi adjacency list
  void inisialisasiAdjList(int jumlah_titik)
  {
    adjacency_list = new list<int>[jumlah_titik];
  }

  // Fungsi untuk inisialisasi adjacency matrix
  void inisialisasiAdjMatrix(int jumlah_titik)
  {
    adjacency_matrix = new int *[jumlah_titik];
    for (int i = 0; i < jumlah_titik; i++)
    {
      adjacency_matrix[i] = new int[jumlah_titik];
      for (int j = 0; j < jumlah_titik; j++)
      {
        adjacency_matrix[i][j] = 0; // Inisialisasi matriks
dengan nilai 0(tidak ada edge)
      }
    }
  }

  // Fungsi untuk menambahkan koneksi dari titik awal ke tujuan
  void tambahLintasan(int titik_awal, int titik_tujuan)
  {
    // Update adjacency list
    adjacency_list[titik_awal].push_back(titik_tujuan);

    // Update adjacency matrix
    adjacency_matrix[titik_awal][titik_tujuan] = 1;
    adjacency_matrix[titik_tujuan][titik_awal] = 1;
  }

  // Fungsi untuk menampilkan adjacency list
  void tampilkanAdjList()
```

```
  {
    list<int>::iterator i;

    for (int v = 0; v < jumlah_titik; v++)
    {
      cout << v << " -> ";
      for (i = adjacency_list[v].begin(); i !=
adjacency_list[v].end(); ++i)
      {
        cout << (*i);
        if (next(i, 1) != adjacency_list[v].end())
        {
          cout << " -> ";
        }
      }
      cout << endl;
    }
  }

  // Fungsi untuk menampilkan adjacency matrix
  void tampilkanAdjMatrix()
  {
    for (int i = 0; i < jumlah_titik; i++)
    {
      for (int j = 0; j < jumlah_titik; j++)
      {
        cout << adjacency_matrix[i][j] << " ";
      }
      cout << endl;
    }
  }
};

int main()
{
  cout << "Peta Rumah" << endl;
  int jumlah_titik = 10;

  // Ini "Object petaKu"
  // Object adalah instansiasi dari Class
  Peta petaKu(jumlah_titik);

  // Mendefinisikan data untuk graf Peta
```

```
  petaKu.tambahLintasan(0,1);
  petaKu.tambahLintasan(1,2);
  petaKu.tambahLintasan(1,6);
  petaKu.tambahLintasan(2,3);
  petaKu.tambahLintasan(3,4);
  petaKu.tambahLintasan(4,5);
  petaKu.tambahLintasan(4,7);
  petaKu.tambahLintasan(5,9);
  petaKu.tambahLintasan(5,10);
  petaKu.tambahLintasan(6,5);
  petaKu.tambahLintasan(7,8);

  cout << endl;
  cout << "Adjacency List" << endl;
  petaKu.tampilkanAdjList();

  cout << endl;
  cout << "Adjacency Matrix" << endl;
  petaKu.tampilkanAdjMatrix();
}
```

3. Tampilkan hasil adjacency listnya (5 poin)

```
Adjacency List
0 -> 1
1 -> 2 -> 6
2 -> 3
3 -> 4
4 -> 5 -> 7
5 -> 9 -> 10
6 -> 5
7 -> 8
8 ->
9 ->
10 ->
```

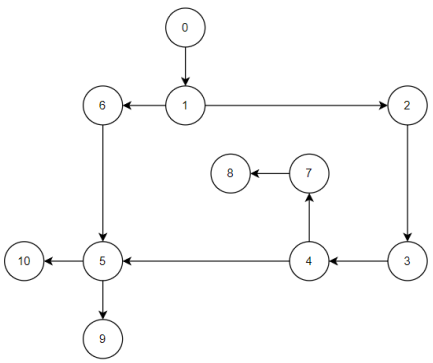
4. Buatkan Class baru bernama "Titik" untuk menyimpan ID titik, nama tempat (misal "Rumah", "Minimarket", "Apotek", dll), titik koordinat x, dan titik koordinat y. Instansiasi class "Titik" untuk menyimpan info titik pada peta. (20 poin)

```
class Titik {
private:
    int id;
    string nama_tempat;
    int x;
    int y;

public:
    Titik(int id, string nama_tempat, int x, int y) : id(id),
nama_tempat(nama_tempat), x(x), y(y) {}

    int getID() { return id; }
    string getNamaTempat() { return nama_tempat; }
    int getX() { return x; }
    int getY() { return y; }
};
```

5. Tampilkan hasil adjacency list berupa nama tempat (15 poin)

```
Adjacency List
Rumah -> pertigaan ->
pertigaan -> Rumah -> kolam renang -> perempatan_gor ->
kolam renang -> pertigaan -> Burger King ->
Burger King -> kolam renang -> pertigaan_Gor ->
pertigaan_Gor -> Burger King -> bunderan_gor -> gor_futsal ->
bunderan_gor -> pertigaan_Gor -> perempatan_gor -> Taman_pinang ->
```

6. Tampilkan hasil graf menggunakan library graphics.h (10 poin)



7. Tambahkan modifikasi lain (1-10 poin)

```
if (petaKu.DFS(titik_awal, titik_tujuan))
    {
        cout << "Path: ";
        petaKu.tampilkanPath();
    }
    else
    {
        cout << "Tidak ada path yang tersedia." << endl;
    }
```

```
    bool DFS(int titik_awal, int titik_tujuan)
    {
        stack<int> s;
        vector<bool> visited(jumlah_titik, false);
        s.push(titik_awal);
        visited[titik_awal] = true;

        while (!s.empty())
        {
            int current = s.top();
            s.pop();
            path.push_back(current);

            if (current == titik_tujuan)
                return true;

            for (int i = 0; i < jumlah_titik; i++)
            {
                if (adjacency_matrix[current][i] == 1 && !visited[i])
                {
                    s.push(i);
                    visited[i] = true;
                }
            }
        }
        return false;
    }
```