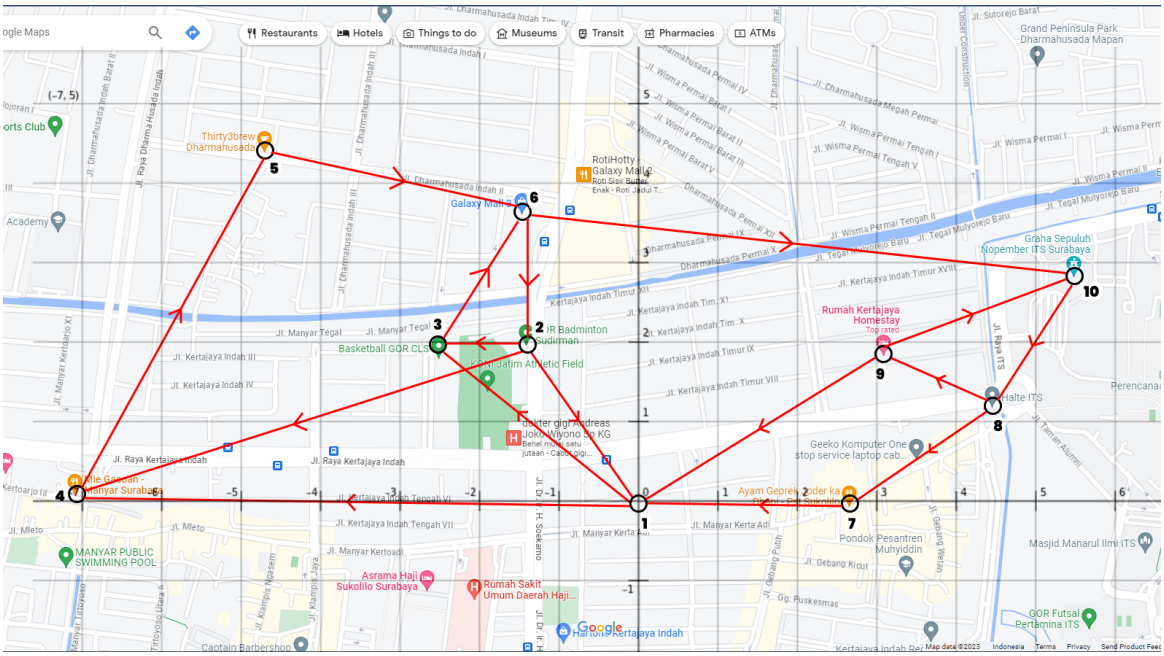
OOP Graph Assignment
Gabriella Erlinda Wijaya - 5027221018

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)
3. Tampilkan hasil adjacency listnya (5 poin)
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)
5. Tampilkan hasil adjacency list berupa nama tempat (15 poin)
6. Tampilkan hasil graf menggunakan library graphics.h (10 poin)
7. Tambahkan modifikasi lain (1-10 poin)

Jawaban:

1. 

2.

```
#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;
    inisialisasiAdjList(jumlah_titik);
    inisialisasiAdjMatrix(jumlah_titik);
  }

  // Destructor
  ~Peta() {
    delete[] adjacency_list;
    for (int i = 0; i < jumlah_titik; i++) {
      delete[] adjacency_matrix[i];
    }
    delete[] adjacency_matrix;
  }


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

  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 - 1].push_back(titik_tujuan - 1);
    // Update adjacency matrix
    adjacency_matrix[titik_awal - 1][titik_tujuan - 1] = 1;
    adjacency_matrix[titik_tujuan - 1][titik_awal - 1] = 1;
  }

  // Fungsi untuk menampilkan adjacency list
  void tampilkanAdjList() {
    list<int>::iterator i;

    for (int v = 0; v < jumlah_titik; v++) {
      cout << v + 1 << " -> ";
      for (i = adjacency_list[v].begin(); i != adjacency_list[v].end();
++i) {
        cout << (*i) + 1;
        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 Kos" << endl;
  int jumlah_titik = 10;

  Peta petaKu(jumlah_titik);

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

```
  //vertex 8
    petaKu.tambahLintasan(8, 7);
    petaKu.tambahLintasan(8, 9);
    petaKu.tambahLintasan(8, 10);
  //vertex 9
    petaKu.tambahLintasan(9, 1);
    petaKu.tambahLintasan(9, 8);
    petaKu.tambahLintasan(9, 10);
  //vertex 10
    petaKu.tambahLintasan(10, 6);
    petaKu.tambahLintasan(10, 8);
    petaKu.tambahLintasan(10, 9);

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

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

**perubahan code = bold text*

Dari codingan yang diberikan pada modul, saya merubah jumlah_titik yang awalnya 5 menjadi 10 sesuai yang diminta soal, lalu alih-alih memulai vertex dari 0 hingga 4, saya memulai vertex dari 1-10. Saya juga memodifikasi bagian //destructor yang mana saya menambahkan for loop untuk memastikan bahwa memori yang digunakan oleh adjacency list dan adjacency matrix dibersihkan ketika objek `Peta` tidak diperlukan lagi. Pada fungsi void tambahLintasan, saya mengurangi semua titik_awal dan titik_tujuan dengan 1 untuk menyesuaikan nomor titik yang saya gunakan adalah mulai dari 1. Hal lain yang saya tambahkan yaitu titik-titik pada `petaKu.tambahLintasan(_, _)` menyesuaikan dari gambar peta rumah saya.

3. Hasil adjacency list

```
Peta Rumah

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

Adjacency Matrix
0 1 1 1 0 0 1 0 1 0
1 0 1 1 0 1 0 0 0 0
1 1 0 0 0 1 0 0 0 0
1 1 0 0 1 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 1 1 0 1 0 0 0 0 1
1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 1 1
1 0 0 0 0 0 0 1 0 1
0 0 0 0 0 1 0 1 1 0
```

4.

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

class Titik {
public:
    int id;
    string nama_tempat;
    double x;
    double y;

    Titik() : id(0), nama_tempat(""), x(0.0), y(0.0) {}
```

```
    Titik(int id, string nama_tempat, double x, double y) {
        this->id = id;
        this->nama_tempat = nama_tempat;
        this->x = x;
        this->y = y;
    }
};

class Peta {
private:
    int jumlah_titik;
    list<int> *adjacency_list;
    int **adjacency_matrix;
    Titik *titik_array;

public:
    Peta(int jumlah_titik) {
        this->jumlah_titik = jumlah_titik;
        inisialisasiAdjList(jumlah_titik);
        inisialisasiAdjMatrix(jumlah_titik);
        titik_array = new Titik[jumlah_titik];
    }

    ~Peta() {
        delete[] adjacency_list;
        for (int i = 0; i < jumlah_titik; i++) {
            delete[] adjacency_matrix[i];
        }
        delete[] adjacency_matrix;
        delete[] titik_array;
    }

    void inisialisasiAdjList(int jumlah_titik) {
        adjacency_list = new list<int>[jumlah_titik];
    }

    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;
            }
        }
    }

    void tambahLintasan(int titik_awal, int titik_tujuan) {
        adjacency_list[titik_awal - 1].push_back(titik_tujuan - 1);
        adjacency_matrix[titik_awal - 1][titik_tujuan - 1] = 1;
        adjacency_matrix[titik_tujuan - 1][titik_awal - 1] = 1;
    }

    void tampilkanAdjList() {
        list<int>::iterator i;

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

    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;
        }
    }

    void tambahTitik(int id, string nama_tempat, double x, double y) {
```

```
        titik_array[id - 1] = Titik(id, nama_tempat, x, y);
    }

    void tampilkanTitik() {
        for (int i = 0; i < jumlah_titik; i++) {
            cout << "Titik: " << titik_array[i].id << endl;
            cout << "Nama Tempat: " << titik_array[i].nama_tempat <<
endl;
            cout << "Koordinat (x, y): (" << titik_array[i].x << ", " <<
titik_array[i].y << ")" << endl;
            cout << endl;
        }
    }
};

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

    Peta petaKu(jumlah_titik);

  //vertex 1
    petaKu.tambahLintasan(1, 2);
    petaKu.tambahLintasan(1, 3);
    petaKu.tambahLintasan(1, 4);
    petaKu.tambahLintasan(1, 7);
    petaKu.tambahLintasan(1, 9);
  //vertex 2
    petaKu.tambahLintasan(2, 1);
    petaKu.tambahLintasan(2, 3);
    petaKu.tambahLintasan(2, 4);
    petaKu.tambahLintasan(2, 6);
  //vertex 3
    petaKu.tambahLintasan(3, 1);
    petaKu.tambahLintasan(3, 2);
    petaKu.tambahLintasan(3, 6);
  //vertex 4
    petaKu.tambahLintasan(4, 1);
    petaKu.tambahLintasan(4, 2);
```

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

    petaKu.tambahTitik(1, "Home", 0.0, 0.0);
    petaKu.tambahTitik(2, "Badminton Court", -1.0, 2.0);
    petaKu.tambahTitik(3, "Basketball Court", -2.0, 2.0);
    petaKu.tambahTitik(4, "Noodle House", -7.0, 0.0);
    petaKu.tambahTitik(5, "Coffee Shop", -5.0, 4.0);
    petaKu.tambahTitik(6, "Mall", -1.0, 4.0);
    petaKu.tambahTitik(7, "Chicken Palace", 3.0, 0.0);
    petaKu.tambahTitik(8, "Bus Stop", 4.0, 1.0);
    petaKu.tambahTitik(9, "Homestay", 3.0, 2.0);
    petaKu.tambahTitik(10, "ITS hall", 6.0, 3.0);

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

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

    cout << endl;
    cout << "Informasi Titik" << endl;
    petaKu.tampilkanTitik();
}
```

Pada code ini saya menambahkan class baru yaitu Titik untuk menyimpan ID titik dan koordinat dari setiap titik yang telah saya gambarkan pada no.1. Saya juga menambahkan `void petaKu.tambahTitik` untuk menambahkan titik dan ID titik sesuai peta yang ada, dan menambahkan koordinat titik x,y. Kemudian untuk menampilkan hasil koordinatnya, saya menambahkan `void tampilkanTitik` yang nantinya akan di print untuk memberikan informasi titik.

5.

```
Informasi Titik
Titik: 1
Nama Tempat: Home
Koordinat (x, y): (0, 0)

Titik: 2
Nama Tempat: Badminton Court
Koordinat (x, y): (-1, 2)

Titik: 3
Nama Tempat: Basketball Court
Koordinat (x, y): (-2, 2)

Titik: 4
Nama Tempat: Noodle House
Koordinat (x, y): (-7, 0)

Titik: 5
Nama Tempat: Coffee Shop
Koordinat (x, y): (-5, 4)

Titik: 6
Nama Tempat: Mall
Koordinat (x, y): (-1, 4)

Titik: 7
Nama Tempat: Chicken Palace
Koordinat (x, y): (3, 0)

Titik: 8
Nama Tempat: Bus Stop
Koordinat (x, y): (4, 1)

Titik: 9
Nama Tempat: Homestay
Koordinat (x, y): (3, 2)

Titik: 10
Nama Tempat: ITS hall
Koordinat (x, y): (6, 3)
```

6.

```
#include <iostream>
#include <list>
#include <string>
#include <graphics.h>
using namespace std;

class Titik {
public:
    int id;
    string nama_tempat;
    double x;
    double y;

    Titik() : id(0), nama_tempat(""), x(0.0), y(0.0) {}

    Titik(int id, string nama_tempat, double x, double y) {
        this->id = id;
        this->nama_tempat = nama_tempat;
        this->x = x;
        this->y = y;
    }
};

class Peta {
private:
    int jumlah_titik;
    list<int> *adjacency_list;
    int **adjacency_matrix;
    Titik *titik_array;

public:
    Peta(int jumlah_titik) {
        this->jumlah_titik = jumlah_titik;
        inisialisasiAdjList(jumlah_titik);
        inisialisasiAdjMatrix(jumlah_titik);
        titik_array = new Titik[jumlah_titik];
    }
```

```
    ~Peta() {
        delete[] adjacency_list;
        for (int i = 0; i < jumlah_titik; i++) {
            delete[] adjacency_matrix[i];
        }
        delete[] adjacency_matrix;
        delete[] titik_array;
    }

    void inisialisasiAdjList(int jumlah_titik) {
        adjacency_list = new list<int>[jumlah_titik];
    }

    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;
            }
        }
    }

    void tambahLintasan(int titik_awal, int titik_tujuan) {
        adjacency_list[titik_awal - 1].push_back(titik_tujuan - 1);
        adjacency_matrix[titik_awal - 1][titik_tujuan - 1] = 1;
        adjacency_matrix[titik_tujuan - 1][titik_awal - 1] = 1;
    }

    void tampilkanAdjList() {
        list<int>::iterator i;

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

```
                }
            }
            cout << endl;
        }
    }

    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;
        }
    }

    void tambahTitik(int id, string nama_tempat, double x, double y) {
        titik_array[id - 1] = Titik(id, nama_tempat, x, y);
    }

    void tampilkanTitik() {
        for (int i = 0; i < jumlah_titik; i++) {
            cout << "Titik: " << titik_array[i].id << endl;
            cout << "Nama Tempat: " << titik_array[i].nama_tempat <<
endl;
            cout << "Koordinat (x, y): (" << titik_array[i].x << ", " <<
titik_array[i].y << ")" << endl;
            cout << endl;
        }
    }
};

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

    int gdriver = DETECT, gmode;
    initgraph(&gdriver, &gmode, "");

    Peta petaKu(jumlah_titik);
```

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

```
  //vertex 10
    petaKu.tambahLintasan(10, 6);
    petaKu.tambahLintasan(10, 8);
    petaKu.tambahLintasan(10, 9);

    petaKu.tambahTitik(1, "Home", 0.0, 0.0);
    petaKu.tambahTitik(2, "Badminton Court", -1.0, 2.0);
    petaKu.tambahTitik(3, "Basketball Court", -2.0, 2.0);
    petaKu.tambahTitik(4, "Noodle House", -7.0, 0.0);
    petaKu.tambahTitik(5, "Coffee Shop", -5.0, 4.0);
    petaKu.tambahTitik(6, "Mall", -1.0, 4.0);
    petaKu.tambahTitik(7, "Chicken Palace", 3.0, 0.0);
    petaKu.tambahTitik(8, "Bus Stop", 4.0, 1.0);
    petaKu.tambahTitik(9, "Homestay", 3.0, 2.0);
    petaKu.tambahTitik(10, "ITS hall", 6.0, 3.0);

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

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

    cout << endl;
    cout << "Informasi Titik" << endl;
    petaKu.tampilkanTitik();

    closegraph();
    return 0;
}
```

Untuk proses run code dengan library graphics.h saya mengalami kendala untuk proses compile dan build code nya.

7.

```
#include <iostream>
#include <list>
```

```
#include <stack>
#include <stdio.h>
#include <string>
#include <math.h>
using namespace std;

class Titik {
public:
    int id;
    string nama_tempat;
    double x;
    double y;

    Titik() : id(0), nama_tempat(""), x(0.0), y(0.0) {}

    Titik(int id, string nama_tempat, double x, double y) {
        this->id = id;
        this->nama_tempat = nama_tempat;
        this->x = x;
        this->y = y;
    }
};

class Peta {
private:
    int jumlah_titik;
    list<int> *adjacency_list;
    int **adjacency_matrix;
    Titik *titik_array;

public:
    Peta(int jumlah_titik) {
        this->jumlah_titik = jumlah_titik;
        inisialisasiAdjList(jumlah_titik);
        inisialisasiAdjMatrix(jumlah_titik);
        titik_array = new Titik[jumlah_titik];
    }

    ~Peta() {
```

```
        delete[] adjacency_list;
        for (int i = 0; i < jumlah_titik; i++) {
            delete[] adjacency_matrix[i];
        }
        delete[] adjacency_matrix;
        delete[] titik_array;
    }

    void inisialisasiAdjList(int jumlah_titik) {
        adjacency_list = new list<int>[jumlah_titik];
    }

    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;
            }
        }
    }

    void tambahLintasan(int titik_awal, int titik_tujuan) {
        adjacency_list[titik_awal - 1].push_back(titik_tujuan - 1);
        adjacency_matrix[titik_awal - 1][titik_tujuan - 1] = 1;
        adjacency_matrix[titik_tujuan - 1][titik_awal - 1] = 1;
    }

    void tampilkanAdjList() {
        list<int>::iterator i;

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

```
            }
            cout << endl;
        }
    }

    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;
        }
    }

    void tambahTitik(int id, string nama_tempat, double x, double y) {
        titik_array[id - 1] = Titik(id, nama_tempat, x, y);
    }

    void tampilkanTitik() {
        for (int i = 0; i < jumlah_titik; i++) {
            cout << "Titik: " << titik_array[i].id << endl;
            cout << "Nama Tempat: " << titik_array[i].nama_tempat << endl;
            cout << "Koordinat (x, y): (" << titik_array[i].x << ", " <<
titik_array[i].y << ")" << endl;
            cout << endl;
        }
    }

    double hitungJarak(int titik_awal, int titik_tujuan) {
        if (titik_awal < 1 || titik_awal > jumlah_titik || titik_tujuan <
1 || titik_tujuan > jumlah_titik) {
            cout << "ID titik tidak valid." << endl;
            return -1.0; // Nilai negatif menunjukkan kesalahan
        }

        double x1 = titik_array[titik_awal - 1].x;
        double y1 = titik_array[titik_awal - 1].y;
        double x2 = titik_array[titik_tujuan - 1].x;
        double y2 = titik_array[titik_tujuan - 1].y;
```

```
        // Menghitung jarak dengan rumus Euclidean distance
        double jarak = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 -
y1));

        return jarak;
    }
};

int main() {
    cout << "Peta Rumah" << endl;
    int jumlah_titik = 10;
    int titik_awal, titik_tujuan;
    int choice;

    Peta petaKu(jumlah_titik);

  //vertex 1
    petaKu.tambahLintasan(1, 2);
    petaKu.tambahLintasan(1, 3);
    petaKu.tambahLintasan(1, 4);
    petaKu.tambahLintasan(1, 7);
    petaKu.tambahLintasan(1, 9);
  //vertex 2
    petaKu.tambahLintasan(2, 1);
    petaKu.tambahLintasan(2, 3);
    petaKu.tambahLintasan(2, 4);
    petaKu.tambahLintasan(2, 6);
  //vertex 3
    petaKu.tambahLintasan(3, 1);
    petaKu.tambahLintasan(3, 2);
    petaKu.tambahLintasan(3, 6);
  //vertex 4
    petaKu.tambahLintasan(4, 1);
    petaKu.tambahLintasan(4, 2);
    petaKu.tambahLintasan(4, 5);
  //vertex 5
    petaKu.tambahLintasan(5, 4);
    petaKu.tambahLintasan(5, 6);
```

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

  petaKu.tambahTitik(1, "Home", 0.0, 0.0);
  petaKu.tambahTitik(2, "Badminton Court", -1.0, 2.0);
  petaKu.tambahTitik(3, "Basketball Court", -2.0, 2.0);
  petaKu.tambahTitik(4, "Noodle House", -7.0, 0.0);
  petaKu.tambahTitik(5, "Coffee Shop", -5.0, 4.0);
  petaKu.tambahTitik(6, "Mall", -1.0, 4.0);
  petaKu.tambahTitik(7, "Chicken Palace", 3.0, 0.0);
  petaKu.tambahTitik(8, "Bus Stop", 4.0, 1.0);
  petaKu.tambahTitik(9, "Homestay", 3.0, 2.0);
  petaKu.tambahTitik(10, "ITS hall", 6.0, 3.0);

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

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

```
    cout << endl;
    cout << "Informasi Titik" << endl;
    petaKu.tampilkanTitik();

    while (true) {
        cout << endl;
        cout << "Menu:" << endl;
        cout << "1. Hitung jarak" << endl;
        cout << "2. Exit program" << endl;
        cout << "Input Choice: ";
        cin >> choice;

        if (choice == 1) {
            cout << "Pilih titik awal (ID): ";
            cin >> titik_awal;

            cout << "Pilih titik tujuan (ID): ";
            cin >> titik_tujuan;

            double jarak = petaKu.hitungJarak(titik_awal, titik_tujuan);
            if (jarak >= 0) {
                cout << "Jarak antara titik " << titik_awal << " dan " <<
titik_tujuan << " adalah " << jarak << " satuan." << endl;
            }
        } else if (choice == 2) {
            cout << "Exiting the program." << endl;
            break;  // Keluar dari loop dan program berhenti
        } else {
            cout << "Pilihan tidak valid. Silakan pilih 1 atau 2." <<
endl;
        }
    }
}
```

Pada code ini saya menambahkan fiitur untuk menghitung jarak antar titik yang ada pada peta saya. Saya menggunakan fungsi double yang nantinya akan menghitung jarak menggunakan rumus phytagoras dan me-return hasil perhitungan tersebut. Pada fungsi main, saya menambahkan opsi untuk user apakah ingin menghitung suatu jarak kedua

titik, atau user memilih untuk keluar dari program. Program akan terus berjalan ketika user masih memilih untuk menghitung jarak titik. Ketika user memilih untuk menghitung jarak titik, maka program akan melanjutkan untuk memberi opsi kepada user, mana titik asal dan mana titik tujuan, lalu program akan secara otomatis menjalankan fungsi hitungJarak dan mencetak hasil perhitungannya.

Output: