

Usulan Soal UAS IF230 – Algorithm and Data Structure

Semester Genap 2021 – 2022

Perintah Soal

1. Kerjakan soal-soal di bawah ini dengan **program berekstensi .c**. Program yang dibuat dengan **ekstensi selain yang ada di perintah soal tidak akan dinilai**.

Format : Nama_NIM_Soalxx.c

Contoh :

Untuk soal nomor 1: Daigo_98765_Soal01.c

Untuk soal nomor 2: Daigo_98765_Soal02.c

Pada saat pengumpulan, **jangan lupa untuk menyertakan file .txt** yang diperlukan, seperti :

Format :

Nama_NIM_peta.txt

Nama_NIM_ongkir.txt

Contoh :

Daigo_98765_peta.txt

Daigo_98765_ongkir.txt

Semua *file* terkait UAS (*file* .c, .exe, .txt) disatukan ke dalam sebuah zip dengan format penamaan sebagai berikut.

Format :

Nama_NIM_UAS_(Kelas).zip

Contoh :

Daigo_98765_UAS_AL.zip

2. Untuk mempermudah penilaian, pada **bagian awal program** setelah **deklarasi library** yang digunakan, **sertakan komen** sebagai berikut dan sesuaikan dengan nama dan NIM masing-masing.

```

1  #include <stdio.h>
2  #include <malloc.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #include <conio.h>
6  #include <windows.h>
7  #include <limits.h>
8
9  /*
10     Ini hanyalah contoh, isi dengan nama dan NIM masing-masing
11     Nama : Shigeno Daigo
12     NIM  : 000 000 98765
13  */

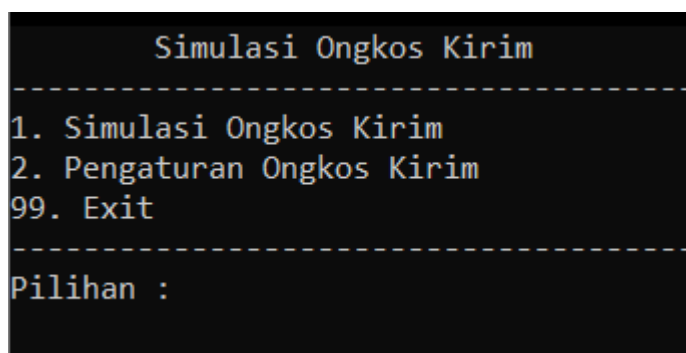
```

Soal 1 (70 poin)

Salah satu masalah utama yang sering dipertimbangkan oleh pembeli dan penjual pada suatu *platform* daring ketika bertransaksi adalah masalah ongkos kirim. Anda sebagai pengembang suatu aplikasi *platform* daring diminta untuk menyimulasikan permasalahan ini secara dinamis dengan data yang tersimpan pada suatu *file* dan diproses menggunakan konsep *file processing* dan *graph Requirement* dari program simulasi ini sebagai berikut.

1. Menu utama program ini memuat **dua fitur utama**, yaitu :
 - a. Simulasi ongkos kirim
 - b. Pengaturan ongkos kirim

Contoh tampilan menu utama dapat dilihat sebagai berikut.



```

Simulasi Ongkos Kirim
-----
1. Simulasi Ongkos Kirim
2. Pengaturan Ongkos Kirim
99. Exit
-----
Pilihan :

```

2. Pada simulasi ongkos kirim, **kode dan daftar kota** dapat **disajikan** dalam bentuk *list* atau *tabel*. Untuk **keluar dari menu simulasi ongkos kirim**, *user* perlu **memasukkan input kode kota asal dan tujuan dengan angka 0**. *User* hanya perlu memasukkan **kode kota asal** dan **tujuan** untuk memunculkan **simulasi ongkos kirim** dengan nama kota yang ada di *file* peta.txt dan matriks *adjacency* yang ada di *file* ongkir.txt.

Beberapa contoh perhitungan simulasi ongkos kirim ada pada gambar di bawah ini.

```

Simulasi Ongkos Kirim
-----
Daftar Kota : (Angka di depan merupakan kode kota)
1. Jakarta
2. Bandung
3. Medan
4. Palembang

Petunjuk :
1. Untuk melakukan simulasi ongkos kirim, masukkan input dalam format kode kota
2. Untuk keluar dari simulasi, masukkan input kode kota asal dan tujuan "0"

Kota Asal      : 1
Kota Tujuan    : 1
Biaya yang dibutuhkan untuk mengirimkan makanan dari Jakarta ke Jakarta adalah Rp. 10000

```

```

Simulasi Ongkos Kirim
-----
Daftar Kota : (Angka di depan merupakan kode kota)
1. Jakarta
2. Bandung
3. Medan
4. Palembang

Petunjuk :
1. Untuk melakukan simulasi ongkos kirim, masukkan input dalam format kode kota
2. Untuk keluar dari simulasi, masukkan input kode kota asal dan tujuan "0"

Kota Asal      : 1
Kota Tujuan    : 3
Biaya yang dibutuhkan untuk mengirimkan makanan dari Jakarta ke Medan adalah Rp. 35000

```

Selama *user* belum memasukkan *input* untuk keluar dari simulasi, maka simulasi akan terus berjalan.

3. *User* dapat memilih untuk **mengatur ulang isi file peta.txt dan ongkir.txt dengan menu pengaturan ongkos kirim**. Jika *file* peta.txt dan ongkir.txt belum diatur, maka *user* dapat langsung diarahkan untuk mengatur kedua *file* ini. Namun, jika sudah pernah ada isi pada *file* peta.txt dan ongkir.txt, maka *user* akan diminta konfirmasi untuk mengatur ulang kedua *file*.

Perhatikan bahwa *file* tidak akan di-rewrite selama *user* belum menyelesaikan konfigurasi *file*.

Ada pun hal-hal yang bisa diatur ulang sebagai berikut.

- a. *User* dapat memasukkan ulang nama kota untuk disimpan pada *file* peta.txt.
- b. *User* dapat memasukkan besaran ongkos kirim dari satu kota ke dirinya sendiri atau ke kota lain.

Berikut adalah contoh tampilan saat mengatur ulang kedua hal ini.

```

Pengaturan Ongkos Kirim
-----
Aplikasi menemukan daftar ongkos kirim yang lama. Apakah Anda yakin ingin menghapus data lama? (Y/N)
Y

```

Pengaturan Ongkos Kirim

Banyaknya kota yang akan diatur simulasi ongkos kirimnya : 4

Kota ke-1

Nama Kota : Palembang

Kota ke-2

Nama Kota : Jakarta

Kota ke-3

Nama Kota : Surabaya

Kota ke-4

Nama Kota : Makassar

```

-----
                          Pengaturan Ongkos Kirim
-----

Perhatikan daftar kota di bawah untuk pengisian data selanjutnya!

Daftar kota :
1. Makassar
2. Bandung
3. Surabaya
4. Makassar

Kota asal : 1
Kota tujuan : 1
Ongkos kirim : Rp. 10000

Kota asal : 1
Kota tujuan : 2
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 14000

Kota asal : 1
Kota tujuan : 3
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 16000

Kota asal : 1
Kota tujuan : 4
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 18000

Kota asal : 2
Kota tujuan : 2
Ongkos kirim : Rp. 7000

Kota asal : 2
Kota tujuan : 3
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 9000

Kota asal : 2
Kota tujuan : 4
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 11000

Kota asal : 3
Kota tujuan : 3
Ongkos kirim : Rp. 15000

Kota asal : 3
Kota tujuan : 4
* Ongkos kirim ini berlaku untuk perjalanan sebaliknya *
Ongkos kirim : Rp. 13000

Kota asal : 4
Kota tujuan : 4
Ongkos kirim : Rp. 17000

Apakah Anda yakin dengan data ini? (Y/N)
Y

```

4. Terdapat dua *file* yang perlu dibuat untuk menyimpan proses simulasi ongkos kirim ini, yaitu:

- NIM_Nama_peta.txt** yang berisi **n** baris. Setiap baris menyatakan **nama kota dan ID kota** tersebut pada program simulasi.
- NIM_Nama_ongkir.txt** yang berisi **n^2** baris. Setiap baris menyatakan **besaran ongkos kirim** yang nantinya akan **dikonversi menjadi matriks *adjacency*** untuk setiap kota.

Usahakan tidak menggunakan *delimiter* # (tanda pagar/hashtag) pada *file* .txt ini.

Isilah data pada *peta.txt* dan *ongkir.txt* dengan data *default* yang kalian inginkan. Pastikan bahwa isi data *peta.txt* dan *ongkir.txt* tepat dan dapat digunakan.

Contoh *file* *peta.txt* dan *ongkir.txt* dapat dilihat di bawah ini.

File *peta.txt*

```
Jakarta || 0
Bandung || 1
Medan || 2
Palembang || 3
```

File *ongkir.txt*

```
10000
40000
50000
25000
40000
5000
38000
17000
50000
38000
5000
10000
25000
17000
10000
7000
```

Penjelasan:

File ongkir.txt memuat ongkos kirim antar kota. Jika *n* merupakan banyaknya kota yang ada, maka **baris ke 1 sampai *n*** memuat **ongkos kirim kota pertama dengan dirinya sendiri dan kota-kota lain**, **baris ke *n*+1 sampai *2n*** memuat **ongkos kirim kota kedua dengan dirinya sendiri dan kota-kota lain**, dan seterusnya.

Rubrik Penilaian Soal 1

Aspek	Kriteria Penilaian				
	Sangat Kurang	Kurang	Cukup	Baik	Sangat Baik
	0%-15%	16%-35%	36%-55%	56%-85%	86%-100%
a. Kelengkapan Fitur (15)	Tidak ada fitur yang diselesaikan sama sekali.	Terdapat kedua fitur, tetapi tidak tepat.	Terdapat salah satu fitur yang berjalan dengan tepat dan yang lainnya tidak tepat.	Terdapat salah satu fitur yang berjalan dengan tepat dan yang lainnya hampir tepat.	Kedua fitur berjalan dengan tepat.
b. Kelengkapan data (7)	Tidak membuat kedua <i>file</i> .txt.	Membuat <i>file</i> .txt, tetapi tidak berkaitan dengan program.	Membuat salah satu <i>file</i> .txt dan kosong.	Membuat kedua <i>file</i> .txt dan kosong.	Membuat kedua <i>file</i> .txt beserta isinya.
c. Konsep <i>graph</i> (10)	Tidak ada konsep <i>graph</i> yang diterapkan.	Ada konsep <i>struct</i> yang diterapkan dan tidak mengarah pada <i>graph</i> .	Ada konsep <i>graph</i> yang diterapkan, tetapi sebagian besar salah.	Ada konsep <i>graph</i> yang diterapkan dan hampir benar.	Konsep <i>graph</i> yang diterapkan tepat.
d. <i>File processing</i> (15)	Tidak ada kode terkait <i>file processing</i> .	Ada kode terkait <i>file processing</i> , tetapi salah fatal (sintaks, dan sebagainya).	Ada kode terkait <i>file processing</i> , tetapi tidak mampu mengolah data eksternal.	Ada kode terkait <i>file processing</i> yang hampir benar.	Kode terkait <i>file processing</i> berjalan sempurna.
e. <i>Error Handling</i> (8)	Tidak membuat	Membuat <i>error handling</i> tidak	Membuat <i>error handling</i> tidak konsisten, tetapi	Membuat <i>error handling</i> konsisten, tetapi	Membuat <i>error handling</i> konsisten

	<i>error handling.</i>	konsisten dan kalimatnya tidak sesuai konteks.	kalimatnya sesuai konteks.	kalimatnya tidak sesuai konteks.	dan kalimatnya sesuai konteks.
f. UI (10)	Penilaian terkait tampilan aplikasi, mulai dari <i>layout</i> , kerapihan, dan unsur lainnya.				
g. UX (5)	Penilaian terkait posisi opsi <i>input</i> dari <i>user</i> .				

Soal 2 (30 poin)

Pasangan bilangan disebut “berjodoh” jika satuannya dari penjumlahan kedua bilangan ini mencapai suatu bilangan n tertentu yang dimasukkan oleh pengguna. Dengan menggunakan konsep *hashing* terutama *separate chaining*, buatlah sebuah program yang akan mencari kombinasi (**bukan permutasi**) dari bilangan-bilangan *input* dari *user* yang “berjodoh”. Perhatikan bahwa **batasan-batasan program** sebagai berikut.

1. Pertama-tama, **program akan menerima satuan** dari **jumlah bilangan** yang disebut “berjodoh” (n). **Ingat bahwa satuan hanya berupa bilangan 0 sampai 9 saja.**
2. Selanjutnya, **program akan menerima sekumpulan bilangan positif** yang akan “dijodohkan”. Lakukan *hashing* pada bilangan-bilangan yang diterima dengan metode *separate chaining*. Perhatikan bahwa *input* berulang hanya boleh masuk ke *chain* sekali saja. Program akan terus menerima bilangan untuk “dijodohkan” selama belum ada *input* bilangan negatif.
3. Terakhir, **program akan menampilkan kombinasi bilangan yang “berjodoh”** dari **hasil perhitungan sebelumnya**. Perhatikan bahwa kombinasi yang ditampilkan **tidak boleh reverse** atau **kembar**.

Contoh **kombinasi reverse** sebagai berikut.

(44, 34) dan (34,44) => cukup **pilih salah satu** untuk ditampilkan di program.

Contoh **kombinasi kembar** sebagai berikut.

(44, 44), (34, 34) => **tidak boleh ditampilkan** sama sekali.

Berikut contoh tampilan program.

```
Selamat datang di aplikasi penjodoh bilangan
Silakan masukan satuan dari jumlah bilangan yang "berjodoh" : 9
Masukkan bilangan ke-1 : 013
Masukkan bilangan ke-2 : 138918
Masukkan bilangan ke-3 : 318731
Masukkan bilangan ke-4 : 4617641
Masukkan bilangan ke-5 : 631736
Masukkan bilangan ke-6 : -1

Daftar bilangan yang berjodoh sebagai berikut.
Pasangan ke-1 : (318731, 138918)
Pasangan ke-2 : (4617641, 138918)
Pasangan ke-3 : (13, 631736)

-----
Process exited after 7.961 seconds with return value 0
Press any key to continue . . .
```

Rubrik Penilaian Soal 2

Aspek	Kriteria Penilaian				
	Sangat Kurang	Kurang	Cukup	Baik	Sangat Baik
	0%-15%	16%-35%	36%-55%	56%-85%	86%-100%
a. Kelengkapan Fitur (10)	Tidak ada program yang dikumpulkan.	Hanya menerima <i>input</i> satuan.	Menerima <i>input</i> satuan dan kumpulan bilangan positif.	Program hampir jadi dengan sedikit <i>bug</i> . (Contoh: program tidak berakhir dengan <i>return value</i> 0).	Program berfungsi dengan baik.
b. Konsep Hashing (10)	Tidak menggunakan <i>hashing</i> .	Ada usaha menggunakan konsep <i>hashing</i> .	Metode <i>hashing</i> bukan <i>separate chaining</i> .	Menggunakan <i>hashing separate chaining</i> yang sebagian besar.	Menggunakan <i>hashing separate chaining</i> dengan tepat.

c. Hasil (7)	Hasil tidak tepat sama sekali.	Hasil sebagian kecil benar.	Hasil hampir benar.	Hasil benar tetapi mengandung <i>reverse</i> atau kembar.	Hasil benar.
d. Kerapian tampilan (3)	Tidak rapi.	Kurang rapi.	Cukup rapi.	Rapi.	Sangat rapi.