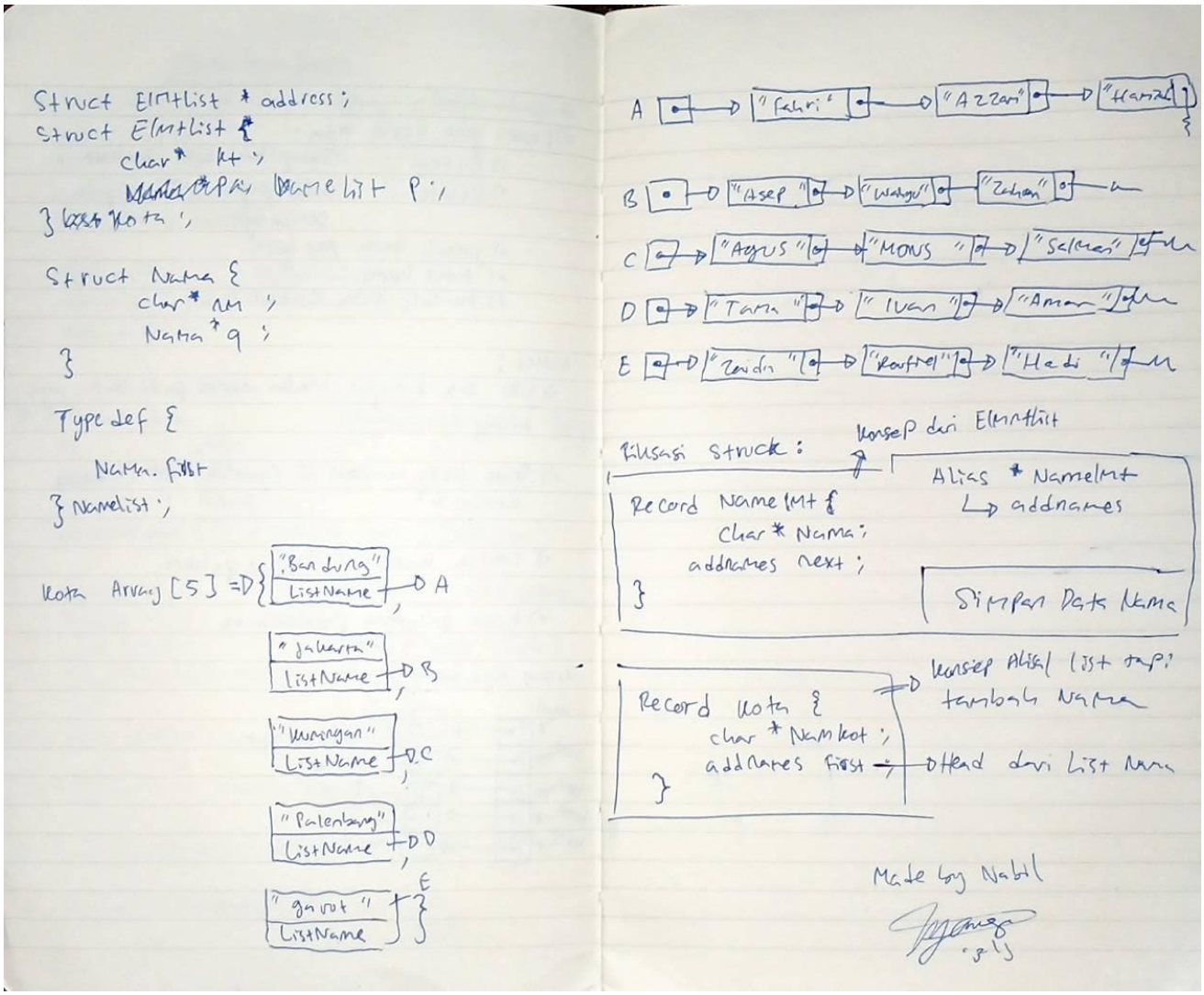
PENJELASAN KASUS 5

Studi Kasus 5: Membuat sebuah program yang mengelola nama – nama yang ada pada suatu Kumpulan kota yang memenuhi requirement berikut:

1. User dapat mengentrykan kota baru (selama tidak melebihi batas kota/Array)
2. Ketika user menghapus suatu kota maka data nama pada kota tersebut juga perlu dihapus
3. User dapat menghapus nama pada suatu kota
4. Program dapat menampilkan daftar nama disalah satu kota ataupun seluruh kota

1.1 Gambaran structur data:



1.2 Komponen ADT yang akan digunakan:

1. Linked.h (sudah dimodifikasi agar bisa menerima char*)
2. Linked.c (implementasi program linked.h) [#include “linked.h”]
3. Main.c (main program) [#include “logic.h”]

4. CaseCity.h (ADT yang akan menampung kebutuhan tambahan untuk array kota dll) [#include “linked.h”]
5. CaseCity.c (implementasi program dari CaseCity.h) [#include “CaseCity.h”]
6. Ui.h (ADT yang akan menyimpan Ui, dipisahkan sebagai bentuk Latihan bila menggunakn GUI)
7. Ui.c (implementasi program dari ui.h) [#include “ui.h”]
8. Logic.h (ADT logic untuk memenuhi studi kasus) [#include “ui.h”, #include “CaseCity.h”]
9. Logic.c (implementasi program logic.h) [#include “logic.h”]

1.3 Penyelesaian Requirement

❖ Requirement pertama yaitu “User dapat mengentrykan kota baru (selama tidak melebihi batas kota/Array)” akan terselesaikan dengan backlog berikut:

a. Membuat modul InitArray untuk menginisialisasi Array struct CityData (setiap isi array diisi Nil)

```
void InitArray(CityData* A, int size) {
    for (i = 0; i < size; i++) {
        A[i].kt = Nil;
        A[i].p = Nil;
    }
}
```

Modul ini akan digunakan untuk menginisialisasi isi array dengan tipe CityData yang menampung informasi nama kota dan list dari nama – nama yang ada dikota tersebut (konsepnya seperti typedef List pada praktukum sebelumnya namun sekarang ia memiliki nama).

b. Membuat modul addcity untuk memasukan kota baru ke dalam array selama array belum mencapai kapasitas maksimal/overflow

```
// Add a new city to the array
int AddCity(CityData* A, int *index, int maxIndex, char* cityName) {
    // Check if array is full
    if (*index >= maxIndex) {
        return -1;
    }

    int i;
    for (i = 0; i < *index; i++) {
        if (A[i].kt != Nil && strcmp(A[i].kt, cityName) == 0) {
            return -2;
        } // City exits indicators
    }

    // Add the new city
    A[*index].kt = strdup(cityName);
    if (A[*index].kt != Nil) {
        A[*index].p = NULL;
        (*index)++;
        return *index - 1;  // Return the index of the new city
    }

    return -3;  // Memory allocation failed indicator
}
```

Modul addCity berguna untuk menambah data kota yang baru sesuai dengan ketentuan. Pertama program ini akan mengechek terlebih daulu apakah array kota masih bisa menampung nama baru. Setelah itu ia akan mengecek apakah nama kota yang ingin ditambahkan sudah ada di dalam array (untuk menghindari nama kota yang sama). Terakhir ia akan menambahkan data kota yang baru. Jika disuatu bagian terdapat yang kondisinya tidak terpenuhi maka ia akan mereturn nilai sesuai dengan masalahnya. Agar dapat mengelola error atau kesalahan yang ada, maka modul addCity ini tidak akan dipanggil secara langsung melainkan melalui modul handing.

c. Membuat modul handling bila ada kesalahan

```
void handleAddCity(CityData* cities, int* cityCount) {
    char cityName[MAX_CITY_LENGTH];
    readCityName(cityName);

    int cityIndex = AddCity(cities, cityCount, MAXCITIES , cityName);
    if (cityIndex >= 0) {
        displayCityAddedMessage(cityName);
    } else if (cityIndex == -1) {
        displayCityLimitMessage();
    } else if (cityIndex == -2) {
        displayCityExistsMessage(cityName);
    } else {
        displayMemoryErrorMessage();
    }
}
```

Mudul handleAddCity ini adalah program yang akan menghandle proses penambahan nama kota sehingga kemungkinan error dapat diminimalisir. Modul handle ini disimpan di logic.c, logic ini bisa juga disebut kontroler yang akan

mengatur tampilan display (include dari ui.h) dan alur kebutuhan program (include dari CaseCity.h)

- Requirement kedua yaitu “Ketika user menghapus suatu kota maka data nama pada kota tersebut juga perlu dihapus” akan terselesaikan dengan backlog berikut:
  a. Membuat modul yang menghandle penghapusan nama kota

```
// Handle deleting a city
void handleDeleteCity(CityData* cities, int* cityCount) {
    char cityName[MAX_CITY_LENGTH];

    readCityName(cityName);

    int cityIndex = FindCity(cities, *cityCount, cityName);
    if (cityIndex >= 0) {
        DeleteCity(cities, cityCount, cityIndex);
        displayCityDeletedMessage(cityName);
    } else {
        displayCityNotFoundMessage(cityName);
    }
}
```

Modul handle ini akan meminta input nama kota yang akan dihapus melalui modul readCityName lalu dengan modul findCity akan dicari indeks tempat data kota tersebut disimpan, lalu data kota akan dihapus oleh deleteCity sesuai dengan requirement yang telah ditentukan.

Modul handle ini juga disimpan di logic.h, ia berperan sebagai penghubungan antar alur program delete yang ada di CaseCity.h dengan tampilan yang diatur di ui.h

b. Membuat Modul deleteCity yang akan menghapus nama kota

Modul deleteCity berperan sebagai solusi dari requirement hapus nama kota. Modul deleteCity pertama – tama akan mengecek terlebih dahulu indeks yang sebelumnya diberikan oleh findCity apakah masuk dalam domain array tersebut. Setelah dipastikan benar maka program akan terlebih dahulu menghapus data nama – nama yang ada di kota tersebut. Selanjutnya nama kota akan dihapus (sangat penting untuk didealokasikan karena bertipe string literal), setelah data di hapus program akan mengeser isi array sehingga jika ada data yang masuk ia dapat langsung masuk di indeks akhir.

```c
// Delete a city and all its names
void DeleteCity(CityData* A, int *size, int cityIndex) {
    if (cityIndex >= 0 && cityIndex < *size) {
        // Delete all names in the city
        DeleteAll(&(A[cityIndex].p));

        // Free the city name
        if (A[cityIndex].kt != NULL) {
            free(A[cityIndex].kt);
        }


        int i; // Shift all cities after the deleted city
        for (i = cityIndex; i < *size - 1; i++) {
            A[i].kt = A[i + 1].kt;
            A[i].p = A[i + 1].p;
        }

        // Clear the last position
        A[*size - 1].kt = Nil;
        A[*size - 1].p = Nil;

        // Decrease the size
        (*size)--;
    }
}
```

c. Memastikan modul deleteAll tersedia untuk char*

```c
// DeleteAll untuk pointer - harus dimodifikasi agar bekerja untuk char*
void DeleteAll(address *P) {
    address temp;
    while (*P != NULL) {
        DeleteFirst(P, &temp);
        DeAlokasi(temp);
    }
}
```

Penting untuk memastikan deleteAll ada untuk char*, hal ini karena jika tidak tersedia maka data nama tidak akan terhapus dan ini tidak diharapkan karena selain akan memenuhi salah satu bagian memori (karena pake string literal) hal ini juga akan membuat requirement tidak terpenuhi.

❖ Requirement ketiga yaitu “User dapat menghapus nama pada suatu kota” akan terselesaikan dengan backlog berikut:

a. Membuat modul yang menghandle penghapusan nama

```
// Handle deleting a name from a city
void handleDeleteName(CityData* cities, int cityCount) {
    char cityName[MAX_CITY_LENGTH];
    char personName[MAX_NAME_LENGTH];

    readCityName(cityName);
    int cityIndex = FindCity(cities, cityCount, cityName);

    if (cityIndex >= 0) {
        readPersonName(personName);
        DeleteNameFromCity(cities, cityIndex, personName);
        displayNameDeletedMessage(personName, cityName);
    } else {
        displayCityNotFoundMessage(cityName);
    }
}
```

Modul handle ini berperan untuk mengatur proses penghapusan nama di suatu kota, pertama akan meminta input melalui readCityName untuk meminta input nama kota tempat data nama yang ingin dihapus. Setelah itu findCity akan mencari data nama kota, bila ditemukan maka findCity akan memberikan indeks dari data tersebut. Lalu program akan meminta input melalui readPersonName untuk mendapatkan data nama yang ingin dihapus, lalu program akan menghapusnya melalui DeleteNameFromCity, dan menampilkan pesan bahwa data telah dihapus

b. Membuat modul Deletename untuk menghapus nama di kota tertentu

```
// Delete a name from a specific city
void DeleteNameFromCity(CityData* A, int cityIndex, char* nama) {
    if (cityIndex >= 0) {
        address deletedNode = Delete(&(A[cityIndex].p), nama);
        if (deletedNode != NULL) {
            DeAlokasi(deletedNode);
        }
    }
}
```

Modul deleteNameFromCity berperan untuk menghapus suatu nama dalam suatu kota.

c. Memastikan modul untuk menghapus node nama tersedia dan node yang dihapus siap untuk didealokaiskan.setelah mendapat Alamat data dari module delete ia akan mendealokasikan data tersebut.

```
// Delete untuk char* berdasarkan nama
address Delete(address *P, char* nama) {
    address temp = NULL;

    if (*P != NULL) {
        // If the first node matches
        if (strcmp((*P)->info, nama) == 0) {
            DeleteFirst(P, &temp);
            return temp;
        }

        // Search for the node in the rest of the list
        address prev = *P;
        address current = (*P)->next;

        while (current != NULL && strcmp(current->info, nama) != 0) {
            prev = current;
            current = current->next;
        }

        // If found, delete it
        if (current != NULL) {
            prev->next = current->next;
            current->next = NULL;
            return current;
        }
    }
    return NULL;
}
```

Modul ini perlu dipastikan ada untuk penghapusan data nama, module delete ini akan merubah alur dari sebuah list sehingga data yang dihapus tidak terikat lagi dengan list (nextnya di NULLkan dan data yang menunjuk dia pindah menujuk data setelahnya), lalu module delete akan meretun data alamat data yang dihapus agar dapat didealokasikan atau dipindahkan ke list yang lain.

- Requirement keempat yaitu “Program dapat menampilkan daftar nama disalah satu kota ataupun seluruh kota” akan terselesaikan dengan backlog berikut:
    a. Membuat modul yang akan menghandle penampilan data

```
// Handle displaying all data
void handleDisplayAllData(CityData* cities, int cityCount) {
    PrintAllData(cities, cityCount);
}
```

Gambar diatas merupakan module yang menghandle proses untuk menampilkan seluruh data.

```c
// Handle displaying data for a specific city
void handleDisplayCityData(CityData* cities, int cityCount) {
    char cityName[MAX_CITY_LENGTH];

    readCityName(cityName);
    int cityIndex = FindCity(cities, cityCount, cityName);

    if (cityIndex >= 0) {
        PrintCityData(cities, cityIndex);
    } else {
        displayCityNotFoundMessage(cityName);
    }
}
```

Program yang terdapat pada gambar pertama akan menangani proses menampilkan seluruh data sedangkan program yang kedua ini menghandle proses menampilakan suatu data kota. Alurnya dimulai dengan meminta input melalui modul ReadCityName, kemudian memerika apakah ada didalam array data kota, jika ada maka akan diambil indeksnya dan menggunakan printCityData data nama kota tersebut akan ditampilkan. Program ini juga akan mengatasi apabila ternyata data kota tidak ada.

b. Membuat modul untuk menampilkan data suatu kota

```c
// Print all names in a specific city
void PrintCityData(CityData* A, int cityIndex) {
    if (cityIndex >= 0 && A[cityIndex].kt != NULL) {
        printf("City: %s\n", A[cityIndex].kt);
        printf("Names: ");
        if (A[cityIndex].p != NULL) {
            PrintInfoListC(A[cityIndex].p);
        } else {
            printf("(No names)\n");
        }
    }
}
```

Modul printCityData akan menampilkan data nama yang terdapat pada suatu data kota, apabila data kosong makai akan menampilkan “no names”.

c. Membuat modul untuk menampilkan data seluruh kota

```c
// Print all cities and their names
void PrintAllData(CityData* A, int size) {
    int i;
    printf("=== All Cities and Names ===\n");
    for (i = 0; i < size; i++) {
        if (A[i].kt != NULL) {
            printf("City %d: %s\n", i + 1, A[i].kt);
            printf("Names: ");
            if (A[i].p != NULL) {
                PrintInfoListC(A[i].p);
            } else {
                printf("(No names)\n");
            }
            printf("\n");
        }
    }
}
```

Menampilkan seluruh data nama dari seluruh kota. Alurnya dimulai dengan mengecek terlebih dahulu apakah data kota ada, jika data ada makai a akan menampilkan nama kota diikuti dengan nama nama yang ada didalam kota.

d. Memastikan modul untuk menampilkan list dengan tipe char* tersedia

```c
// Print untuk char*
void PrintInfoListC(address P) {
    address current = P;
    while (current != NULL) {
        printf("%s --> ", current->info);
        current = current->next;
    }
    printf("NULL\n");
}
```

Modul yang akan menampilkan data suatu List bertipe char* (string).