Nama : Yanuar Nurul Hilal
NIM : 222112418
Kelas : 2KS4

Penugasan Struktur Data Praktikum 10

2. Modifikasi program Praktikum10A.c sehingga jumlah data yang disimpan pada tabel hash bisa fleksibel . Simpan hasil modifikasi Anda pada file Praktikum10B.c.
3. Modifikasi program Praktikum10B.c sehingga data pada baris/indeks yang dihapus (flag=2) tidak dapat diisi lagi. Simpan hasil modifikasi Anda pada file Praktikum10C.c.

```c
//Nomor 2 dan 3

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

struct item{
    int key;
    int value;
};

struct hashtable_item{
    int flag;
    /* flag = 0 : Tidak ada data
    * flag = 1 : Ada data
    * flag = 2 : Sebelumnya ada datanya */
    struct item *data;
};

struct hashtable_item *array;
int max;
/* initializing hash table array */
void init_array(){
    int i;
     for (i = 0; i < max; i++)
    {
        array[i].flag = 0;
        array[i].data = NULL;
    }
}

/* to every key, it will generate a corresponding
index */
```

```
int hashcode(int key)
{
     return (key % max);
}

int size = 0; // size dari hashtable
int size_of_hashtable()
{
    return size;
}

void insert(int key, int value)
{
    int index = hashcode(key);
    int i = index;
    /* creating new item to insert in the hash table array */
    struct item *new_item = (struct item*)malloc(sizeof(struct item));
    new_item->key = key;
    new_item->value = value;
    /* probing through the array until we reach an empty space - LINEAR
PROBING*/
    while (array[i].flag == 1)
    {
        if (array[i].data->key == key){
            /* case where already existing key matches the given key */
            printf("\n Key sudah digunakan, value telah diupdate \n");
            array[i].data->value = value;
            return;
            }
        i = (i + 1) % max; //maju satu langkah
        if (i == index) //jika sudah mengecek satu-satu index sampai balik
lagi ke index penuh
        {
            printf("\n Hash telah full, tidak bisa insert \n");
            return;
            }
    }
    if (array[i].flag == 2){   // nomor 3
        printf("Tidak bisa insert data");
    }
    else{
        array[i].flag = 1;
        array[i].data = new_item;
```

```
        size++;
        printf("\n Key (%d) telah dimasukkan \n", key);
    }
}

void remove_element(int key)
{
    int index = hashcode(key);
    int i = index;
    /* probing through array until we reach an empty space where not even
once an element had been present */
    while (array[i].flag != 0){
        if (array[i].flag == 1 && array[i].data->key == key ){
            // case when data key matches the given key
            array[i].flag = 2;
            array[i].data = NULL;
            size--;
            printf("\n Key (%d) has been removed \n",key);
            return;
        }
        i = (i + 1) % max;
        if (i == index){
            break;
        }
    }
    printf("\n Key tidak ada \n");
}

/* to display all the elements of hash table */
void display()
{
    int i;
    for (i = 0; i < max; i++){
        struct item *current = (struct item*)
        array[i].data;
        if (current == NULL)
        {
            printf("\n Array[%d] tidak punya elemen \n",i);
        }
        else
        {
            printf("\n Elemen dari Array[%d] : \n %d(key) dan %d(value) ",
i, current->key, current->value);
```

```
        }
    }
}

int main(){
        int choice, key, value, n, c;
        printf("Masukkan jumlah data : ");scanf("%d",&max);
        array = (struct hashtable_item*) malloc(max *sizeof(struct
hashtable_item));
        init_array();
        do {
            printf("Implementasi Hash Table dengan Linear Probing \n\n");
            printf("MENU: \n1.Insert item pada Hashtable" "\n2.Remove item
dari Hashtable" "\n3.Cek ukuran Hashtable" "\n4.Display Hashtable" "\n\n
Masukkan pilihan:");
            scanf("%d", &choice);
            switch(choice)
            {
             case 1:
                 printf("Insert item pada Hashtable\n");
                 printf("Enter key-:\t");
                 scanf("%d", &key);
                 printf("Enter value-:\t");
                 scanf("%d", &value);
                 insert(key, value);
                 break;

             case 2:
                 printf("Remove item dari Hashtable \nEnter the key yang
ingin didelete-:");
                 scanf("%d", &key);
                 remove_element(key);
                 break;

             case 3:
                 n = size_of_hashtable();
                 printf("Ukuran Hashtable :%d\n", n);
                 break;

             case 4:
                 display();
                 break;
                 default:
```

```
                printf("Input tidak valid\n");
            }
             printf("\n Apakah ingin melanjutkan?:(tekan 1 jika YA)\t");
             scanf("%d", &c);
        }while(c == 1);
    getchar();
    return 0;
}
```

4. Modifikasi program Praktikum10C.c dengan tanpa menggunakan pointer dan collision resolutionnya dengan menggunakan metode quadratic probing. Simpan hasil modifikasi Anda pada file Praktikum10D.c

```
#include <stdio.h>

struct item
{
    int key;
    int value;
};

struct hashtable_item
{
    int flag;
    /* flag = 0 : Tidak ada data
     * flag = 1 : Ada data
     * flag = 2 : Sebelumnya ada datanya */
    struct item data;
};

int max;

/* initializing hash table array */
void init_array(struct hashtable_item array[])
{
    int i;
    for (i = 0; i < max; i++)
    {
        array[i].flag = 0;
```

```
    }
}

int hashcode(int key)
{
    return (key % max);
}

int size = 0; /* size dari hashtable*/

int size_of_hashtable()
{
    return size;
}

void insert(struct hashtable_item array[], int key, int value)
{
    int index = hashcode(key);
    int i = index;
    int j = 1;

    /* creating new item to insert in the hash table array */ struct item
new_item;
    new_item.key = key;
    new_item.value = value;

    /* probing through the array until we reach an empty space - Quadratic
PROBING*/
    while (array[i].flag == 1)
    {
        if (array[i].data.key == key)
        {
            /* case where already existing key matches the given key */
            printf("\n Key telah digunakan, memperbarui value \n");
            array[i].data.value = value;
            return;
        }
        if (array[i].flag == 2)
        {
            printf("\n Indeks ke %d memiliki flag 2", i);
            return;
        }
```

```
        i = (index + (j * j)) % max; // quadratic probing
        j++;
        if (i == index) // jika sudah mengecek satu - satu index sampai
balik lagi ke index penuh
        {
            printf("\n Hash table telah full, tidak bisa insert item \n");
            return;
        }
    }
    array[i].flag = 1;
    array[i].data = new_item;
    size++;
    printf("\n Key (%d) telah ditambahkan \n", key);
}

void remove_element(struct hashtable_item array[], int key)
{
    int index = hashcode(key);
    int i = index;
    int j = 1;
    /* probing through array until we reach an empty
    space where not even once an element had been present
    */
    while (array[i].flag != 0)
    {
        if (array[i].flag == 1 && array[i].data.key == key)
        {
            // case when data key matches the given key
            array[i].flag = 2;
            size--;
            printf("\n Key (%d) telah dihapus \n", key);
            return;
        }
        i = (index + (j * j)) % max;
        j++;
        if (i == index)
        {
            break;
        }
    }
    printf("\n Key tersebut tidak ada \n");
}
```

```c
/* to display all the elements of hash table */
void display(struct hashtable_item array[])
{
    int i;
    for (i = 0; i < max; i++)
    {
        if (array[i].flag != 1)
        {
            printf("\n Array[%d] tidak memiliki elemen \n", i);
        }
        else
        {
            printf("\n Array[%d] mempunyai elemen -: \n %d(key) dan
%d(value) ", i, array[i].data.key, array[i].data.value);
        }
    }
}

int main()
{
    int choice, key, value, n, c;

    printf("Masukkan ukuran Array : ");
    scanf("%d", &max);

    struct hashtable_item array[max];

    init_array(array);
    do
    {
        printf("Implementasi Hash Table dengan Quadratic Probing\n\n");

        printf("Menu : \n1.Insert item pada Hashtable \n2.Delete item dari
Hashtable \n3.Cek ukuran dari Hashtable \n4.Display Hashtable\n\nPilihan
Anda ? : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Insert item pada Hashtable\n");
            printf("Masukan key:\t");
            scanf("%d", &key);
            printf("Masukkan value:\t");
```

```
            scanf("%d", &value);
            insert(array, key, value);
            break;

        case 2:
            printf("Delete item dari Hashtable \n Masukkan key yang akan
dihapus:");
            scanf("%d", &key);
            remove_element(array, key);
            break;

        case 3:
            n = size_of_hashtable();
            printf("Ukuran dari Hashtable adalah:%d\n", n);
            break;
        case 4:
            display(array);
            break;
        default:
            printf("Pilihan tidak valid\n");
        }
        printf("\nApakah ingin melanjutkan ?: (tekan 1 untuk YA)\t");
        scanf("%d", &c);
    } while (c == 1);

    getchar();
    return 0;
}
```