LAMPIRAN

```
clc; clear; close all;
i = imread('data/DATA B/orange2.jpg');

%% CARA %% Imadjust with RGB

i3 = i;

constr = imadjust(i3,[.8 .8 .8;.9 .9 .9]);
gray = rgb2gray(constr);
imcom = imcomplement(gray);

figure;
subplot(1, 2, 1), imshow(imcom), title('Hasil Complemented');
subplot(1, 2, 2), imshow(imcom), title('Hasil OCR');

ocrResults = ocr(i3);
recognizedText3 = ocrResults.Text;
disp(recognizedText3);

%% Boxing %%
features = ocrResults.CharacterBoundingBoxes;
row = max(size(recognizedText3));
i = 1;
for m = 1 : row
    if recognizedText3(i) ~= ' '
        pos = features(m, :);
        rectangle(  'Position',pos, ...
                    'EdgeColor','r', ...
                    'LineWidth', 2.0);
    end
    i = i + 1;
end



%%capture

#@title Import Only

import re
import csv
%%capture

#@title Input Word

# Mengambil input dari pengguna
kata_awal = "OPANQE" #@param {type :"string"}
```

```
# Menampilkan list
print("List yang dibuat:", kata_awal)
%%capture

#@title Input word total

# Menghitung total huruf per kata
auto_perhuruf = len(kata_awal)

input_total = 6 #@param {type :"integer"}

# Input total huruf yang diinginkan
total_huruf = int(input_total) if input_total else
auto_perhuruf
print(total_huruf)
%%capture

#@title Word Cleaner

def bersihkan_karakter_khusus(input_string):
    # Hanya menyisakan huruf (A-Z, a-z) dan menghapus karakter
khusus
    cleaned_string = re.sub(r'[^a-zA-Z]', ' ', input_string)
    return cleaned_string

# Panggil fungsi untuk membersihkan karakter khusus dari input
pengguna
teks_bersih = bersihkan_karakter_khusus(kata_awal)

# Tampilkan hasil
print("Teks setelah dibersihkan: ", teks_bersih)
%%capture

#@title Capital Word Converter

def capt_kata(input_string):
    return input_string.capitalize()

# Panggil fungsi untuk mengonversi ke huruf kapital untuk
huruf awal dalam kata
hasil_konversi = capt_kata(teks_bersih)

# Tampilkan hasil
print("Hasil konversi menjadi huruf kecil:", hasil_konversi)
```

```
#@title Database

# Database nama orang
database_nama = ['Acai', 'Ackee', 'Akebia', 'Almond', 'Amla',
'Annona', 'Apple', 'Apricot', 'Arhat', 'Avocado', 'Banana',
'Barberry', 'Berry', 'Bignay', 'Bilberry', 'Bilimbi', 'Black
sapote', 'Blackberry', 'Blackcurrant', 'Blood orange',
'Blueberry', 'Boysenberry', 'Breadfruit', "Buddha's hand",
'Burahol', 'Cactus pear', 'Cantaloupe', 'Cempedak', 'Chayote',
'Cherry', 'Cloudberry', 'Coco de mer', 'Coconut', 'Cranberry',
'Cucumber', 'Currant', 'Custard apple', 'Date', 'Dragonfruit',
'Duku', 'Durian', 'Eggfruit', 'Eggplant', 'Elderberry',
'Elephant apple', 'Emu apple', 'Feijoa', 'Fig', 'Finger lime',
'Gac', 'Gandaria', 'Genip', 'Goji berry', 'Gooseberry',
'Goumi', "Governor's plum", 'Grape', 'Grapefruit',
'Grumichama', 'Guava', 'Hala fruit', 'Honeyberry', 'Honeydew',
'Huckleberry', 'Imbe', 'Indian fig', 'Jabuticaba',
'Jackfruit', 'Jambul', 'Jelly palm', 'Jujube', 'Kandis',
'Kepel fruit', 'Ketembilla', 'Kiwano', 'Kiwi', 'Kiwifruit',
'Korlan', 'Kumquat', 'Kundong', 'Langsat', 'Lemon',
'Lillypilly', 'Lime', 'Lingonberry', 'Longan', 'Loquat',
'Lychee', 'Mamey sapote', 'Mamoncillo', 'Mangaba', 'Mango',
'Mangosteen', 'Maypop', 'Melon', 'Miracle fruit', 'Monstera
deliciosa', 'Monstera fruit', 'Mountain soursop', 'Mulberry',
'Muscadine', 'Nance', 'Nannyberry', 'Nectarine', 'Neem fruit',
'Noni', 'Olallieberry', 'Orange', 'Papaya', 'Paradise nut',
'Passionfruit', 'Peach', 'Pear', 'Pepino', 'Persimmon',
'Peruvian apple cactus', 'Pindaíba', 'Pineapple', 'Pitahaya',
'Pitanga', 'Plum', 'Plumcot', 'Pomegranate', 'Pulasan',
'Purple mangosteen', 'Quince', 'Rambutan', 'Raspberry',
'Redcurrant', 'Salak', 'Salal berry', 'Sapodilla', 'Satsuma',
'Soursop', 'Spanish lime', 'Starfruit', 'Strawberry', 'Surinam
cherry', 'Tamarillo', 'Tamarind', 'Tangerine', 'Tomato',
'Ume', 'Uvaia', 'Watermelon', 'Wax apple', 'White currant',
'White sapote', 'Yuzu', 'Zalzalak', 'Zigzag vine fruit']
%%capture

#@title First Search Algorithm

# Fungsi untuk mencari nama dalam database yang sesuai dengan
total huruf dan pecahan huruf kata yang dicari
def cari_nama_dalam_database(database, total_huruf,
hasil_konversi):
    nama_terpilih = []
    for nama in database:
```

```
        if len(nama) == total_huruf and hasil_konversi in
nama:
            nama_terpilih.append(nama)
    return nama_terpilih

# Mencari nama-nama yang sesuai dengan kriteria
hasil1 = cari_nama_dalam_database(database_nama, total_huruf,
hasil_konversi)

# Menampilkan nama-nama yang sesuai
if hasil1:
    print("Nama-nama yang memiliki", total_huruf, "huruf dan
mengandung pecahan huruf", hasil_konversi, ":", hasil1)
else:
    print("Tidak ada nama yang sesuai dengan kriteria dalam
database.")
%%capture

#@title Second Search Algorithm

# Fungsi untuk mencari nama dalam database yang sesuai dengan
total huruf dan pecahan huruf kata yang dicari
def cari_nama_dalam_database(database, total_huruf,
hasil_konversi):
    nama_terpilih = []
    for nama in database:
        if len(nama) == total_huruf:
            kata_cocok = True
            for huruf in hasil_konversi:
                if huruf not in nama:
                    kata_cocok = False
                    break
            if kata_cocok:
                nama_terpilih.append(nama)
    return nama_terpilih

# Mencari nama-nama yang sesuai dengan kriteria
hasil2 = cari_nama_dalam_database(database_nama, total_huruf,
hasil_konversi)

# Menampilkan nama-nama yang sesuai
if hasil2:
    print("Nama-nama yang memiliki", total_huruf, "huruf dan
mengandung pecahan huruf", hasil_konversi, ":", hasil2)
else:
```

```
    print("Tidak ada nama yang sesuai dengan kriteria dalam
database.")
%%capture

#@title Third Search Algorithm

# Fungsi untuk mengecek apakah kata memiliki perbedaan
maksimal beberapa huruf tergantung yang batas yang dipilih
def perbedaan_maksimal(kata1, kata2):
    perbedaan = sum(a != b for a, b in zip(kata1, kata2))
    return perbedaan <= len(kata_awal) - 1

# Fungsi untuk mencari kata dalam database dengan total huruf
dan huruf acak
def cari_kata_dalam_database(database, total_huruf,
hasil_konversi):
    kata_terpilih = [
        kata for kata in database
        if len(kata) == total_huruf and
perbedaan_maksimal(kata, hasil_konversi)
    ]
    return kata_terpilih

# Mencari kata yang memiliki jumlah huruf sesuai dengan input
total huruf
# dan memiliki perbedaan maksimal 1 huruf dengan huruf acak
hasil3 = cari_kata_dalam_database(database_nama, total_huruf,
hasil_konversi)

# Menampilkan kata-kata yang sesuai
if hasil3:
    print(f"Kata yang sama dengan {hasil_konversi}: {hasil3}")
else:
    print(f"Tidak ada kata yang sesuai dengan kriteria dalam
database.")
%%capture

#@title Merge All Search Output

# Menggabungkan semua hasil
hasil_total = list(set(hasil1 + hasil2 + hasil3))
hasil_total.sort(key=lambda x: x[0])
print(hasil_total)

# Menghitung total kata dalam hasil_total
```

```
total_kata = len(hasil_total)
print(f"Total kata dalam list: {total_kata}")
%%capture

#@title Highest Percentage Output

file_path = "hasil_persamaan_tertinggi.csv"

def hitung_persamaan(hasil_konversi, hasil_total, file_path,
total_huruf):
    # List untuk menyimpan hasil dengan persentase tertinggi
    hasil_dengan_persentase_tertinggi = []

    # Menghitung persentase tertinggi
    persentase_tertinggi = 0

    for kata in hasil_total:
        # Menghitung panjang kata dalam list
        panjang_kata = len(kata)

        # Menghitung jumlah huruf yang sama dari kata awal dan
kata dalam list
        huruf_sama = sum(1 for a, b in zip(hasil_konversi,
kata) if a == b)

        # Menghitung persentase persamaan huruf
        persentase = (huruf_sama / total_huruf) * 100

        # Menghitung Nilai Hamming Distance
        hamming = total_huruf - huruf_sama

        # Memeriksa apakah persentase saat ini lebih tinggi
dari yang sebelumnya
        if persentase > persentase_tertinggi:
            # Jika ya, update nilai persentase tertinggi dan
hasil dengan persentase tertinggi
            persentase_tertinggi = persentase
            hasil_dengan_persentase_tertinggi = [(kata,
persentase_tertinggi, hamming)]
        # Menambahkan hasil dengan persentase yang sama dengan
persentase tertinggi
        elif persentase == persentase_tertinggi:
            hasil_dengan_persentase_tertinggi.append((kata,
persentase_tertinggi, hamming))
```

```
    # Mengurutkan hasil berdasarkan nilai Hamming Distance dan
abjad
    hasil_dengan_persentase_tertinggi =
sorted(hasil_dengan_persentase_tertinggi, key=lambda x: (x[2],
x[0]))

    # Menyimpan hasil ke csv
    with open(file_path, mode='a+', newline='') as file:
        writer = csv.writer(file)
        if file.tell() == 0: # Periksa jika file kosong
            writer.writerow(["Input Kata","Hasil Kata",
"Tingkat Kemiripan (%)", "Jarak Hamming"])
        for hasil in hasil_dengan_persentase_tertinggi:
            writer.writerow([hasil_konversi ,hasil[0],
hasil[1], hasil[2]])

    # Menampilkan hasil
    print("Hasil pengurutan berdasarkan Hamming Distance:")
    for hasil in hasil_dengan_persentase_tertinggi:
        print(f"Kata: {hasil[0]} dengan tingkat kemiripan
{hasil[1]:.2f}% dan jarak Hamming Distance {hasil[2]}")

hitung_persamaan(hasil_konversi, hasil_total, file_path,
total_huruf)
%%capture

#@title  All Percentage Output

file_path = "hasil_persamaan.csv"

def hitung_persamaan(hasil_konversi, hasil_total, file_path,
total_huruf):

    # List untuk menyimpan hasil dengan persentase tertinggi
    hasil_dengan_persentase = []

    for kata in hasil_total:
        # Menghitung panjang kata dalam list
        panjang_kata = len(kata)

        # Menghitung jumlah huruf yang sama dari kata awal dan
kata dalam list
        huruf_sama = sum(1 for a, b in zip(hasil_konversi,
kata) if a == b)
```

```
        # Menghitung persentase persamaan huruf
        persentase = (huruf_sama / total_huruf) * 100

        # Menghitung Nilai Hamming Distance
        hamming = total_huruf - huruf_sama

        # Menambahkan hasil ke dalam list
        hasil_dengan_persentase.append((kata, persentase,
hamming))

    # Mengurutkan hasil berdasarkan nilai Hamming Distance dan
abjad
    hasil_dengan_persentase = sorted(hasil_dengan_persentase,
key=lambda x: (x[2], x[0]))

    # Menyimpan hasil ke csv
    with open(file_path, mode='a+', newline='') as file:
        writer = csv.writer(file)
        if file.tell() == 0: # Periksa jika file kosong
            writer.writerow(["Input Kata","Hasil Kata",
"Tingkat Kemiripan (%)", "Jarak Hamming"])
        for hasil in hasil_dengan_persentase:
            writer.writerow([kata_awal ,hasil[0], hasil[1],
hasil[2]])

    # Menampilkan hasil
    print("Hasil pengurutan berdasarkan Hamming Distance:")
    for hasil in hasil_dengan_persentase:
        print(f"Kata: {hasil[0]} dengan tingkat kemiripan
{hasil[1]:.2f}% dan jarak Hamming Distance {hasil[2]}")

hitung_persamaan(hasil_konversi, hasil_total,file_path,
total_huruf)
```