Sayed Mohammad Fatemi

student number: 40127793

signal project

phase 1

با توجه به توضیحاتی که در فاز اول داده شده کد را تیکه تیکه شرح خواهم داد

سورس کد :

ابتدا کتابخانه های مربوطه را ایمپورت میکنم

```python
import os
import numpy as np
from scipy.fft import fft
from pydub import AudioSegment
```

یک تابع برای پیدا کردن فرکانس غالب بر اساس قدر مطلق دامنه ایجاد کردم

```python
def dominant_frequency(sound):
    data = np.array(sound.get_array_of_samples())
    rate = sound.frame_rate
    size = len(data)
    transform = fft(data)
    freq = np.arange(size) * (rate / size)
    amp = np.abs(transform)
    peak = np.argmax(amp)
    return freq[peak]
```

یک تابع برای لود کردن فایل ها و پیدا کردن دامنه غالب آن ها

```python
def load_and_process_files(folder, file_list):
    processed_data = []
    for file in file_list:
        path = os.path.join(folder, file)
        sound = AudioSegment.from_file(path, format="m4a")
        freq = dominant_frequency(sound)
        processed_data.append([file, freq])
    return processed_data
```

بر اساس ماکسیمم طول زمانی فایل ها و مینمم آن ها و نیز اینکه با ۶/۸ طرف هستیم و نوع نوت ها سه بازه برای نوت ها در نظر گرفتم و در ضمن برای نوع نوت ها نیز چون باید یک پارامتر آستانه فرکانس مشخص میکردیم که حداکثر اختلاف چقدر باشد کلا این را کنار گذاشتم و نوتی را انتخاب کردم که کمترین فاصله فرکانسی را با نوت مورد نظرم داشته باشد بنابراین هم نوع و هم مدت زمان نوت در این تابع مشخص می شود

```python
def classify_note(file, ref_data):
    path = os.path.join(dir_b, file)
```

```
    sound = AudioSegment.from_file(path, format="m4a")
    freq = dominant_frequency(sound)
    length = len(sound.get_array_of_samples()) / sound.frame_rate

    diff = [abs(entry[1] - freq) for entry in ref_data]
    closest = np.argmin(diff)
    matched = ref_data[closest][0]

    # range of durations based on all files
    if length < 0.3:
        note_type = '8th'
    elif 0.3 <= length <= 0.6:
        note_type = '4th'
    else:
        note_type = 'dot4th'

    return [file, matched, length, note_type]
```

این تابع را برای ساخت آهنگ مرحله آخر گذاشتم به این صورت که بر اساس ترتیب نوت ها که برای آهنگ تولدت مبارک ساختیم درون نوت ها میگردد و نوتی مورد نظر را که همانطور که در مراحل قبل بدست آوردیم دارای کمترین اختلاف فرکانسی با نوت موردنظر است و نیز مدت زمان مورد نظر را نیز دارد انتخاب میکند و به ته آهنگ میچسباند و همینطور پیش می رود تا کل آن ساخته شود

```python
def create_final_song(sequence, output_data, output_folder):
    final_sound = AudioSegment.empty()
    for note_name, type_n in sequence:
        for entry in output_data:
            if entry[1] == note_name and entry[3] == type_n:
                path = os.path.join(output_folder, entry[0])
                sound = AudioSegment.from_file(path, format="m4a")
                final_sound += sound
                break
    return final_sound
```

در این مرحله نیز متغییر های مورد نظر و ترتیب نوت های گفته شده برای آهنگ تولدت مبارک و نیز فراخوانی توابع و ذخیره ی آهنگ را داریم

```python
dir_a = '8-named-notes'
dir_b = '26-hbd-notes'

list_a = [f for f in os.listdir(dir_a) if f.endswith('.m4a')]
list_b = [f for f in os.listdir(dir_b) if f.endswith('.m4a')]

ref_data = load_and_process_files(dir_a, list_a)

output_data = [classify_note(file, ref_data) for file in list_b]

# HBD
sequence = [
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('Do_octave2.m4a', 'dot4th'),
```

```
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('Do_octave2.m4a', 'dot4th'),
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('Do_octave2.m4a', '8th'),
    ('Do_octave2.m4a', '4th'),
    ('Si_octave1.m4a', '8th'),
    ('La_octave1.m4a', '4th'),
    ('Si_octave1.m4a', 'dot4th'),
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('Si_octave1.m4a', '4th'),
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('Si_octave1.m4a', 'dot4th'),
    ('Sol_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('La_octave1.m4a', '8th'),
    ('Sol_octave1.m4a', '4th'),
    ('La_octave1.m4a', '8th'),
    ('Si_octave1.m4a', '4th'),
    ('Do_octave2.m4a', 'dot4th')
]

final_sound = create_final_song(sequence, output_data, dir_b)

final_sound.export('HBD.m4a', format="mp4", codec="aac")    # the output will be m4a
print("Song created successfully.")
```