1 Write a program to Implement pattern recognition problems of speech recognition

```
# importing libraries
import speech_recognition as sr
import os
from pydub import AudioSegment
from pydub.silence import split_on_silence
# a function that splits the audio file into chunks
# and applies speech recognition
def silence_based_conversion(path = "alice-medium.wav"):
# open the audio file stored in
# the local system as a wav file.
song = AudioSegment.from_wav(path)
# open a file where we will concatenate
# and store the recognized text
fh = open("recognized.txt", "w+")
# split track where silence is 0.5 seconds
# or more and get chunks
chunks = split_on_silence(song,
# must be silent for at least 0.5 seconds
# or 500 ms. adjust this value based on user
# requirement. if the speaker stays silent for
# longer, increase this value. else, decrease it.
min_silence_len = 500,
1
# consider it silent if quieter than -16 dBFS
# adjust this per requirement
silence_thresh = -16
)
# create a directory to store the audio chunks.
try:
os.mkdir('audio_chunks')
except(FileExistsError):
pass
# move into the directory to
1
# store the audio files.
os.chdir('audio_chunks')
i = 0
# process each chunk
for chunk in chunks:
# Create 0.5 seconds silence chunk
chunk_silent = AudioSegment.silent(duration = 10)
```

```
# add 0.5 sec silence to beginning and
# end of audio chunk. This is done so that
# it doesn't seem abruptly sliced.
audio_chunk = chunk_silent + chunk + chunk_silent
# export audio chunk and save it in
# the current directory.
print("saving chunk{0}.wav".format(i))
# specify the bitrate to be 192 k
audio_chunk.export("./chunk{0}.wav".format(i), bitrate ='192k', format ="wav")
# the name of the newly created chunk
filename = 'chunk'+str(i)+'.wav'
print("Processing chunk "+str(i))
# get the name of the newly created chunk
# in the AUDIO_FILE variable for later use.
file = filename
# create a speech recognition object
r = sr.Recognizer()
2
# recognize the chunk
with sr.AudioFile(file) as source:
# remove this if it is not working
# correctly.
r.adjust_for_ambient_noise(source)
audio_listened = r.listen(source)
try:
# try converting it to text
rec = r.recognize_google(audio_listened)
# write the output to the file.
fh.write(rec+". ")
# catch any errors.
except sr.UnknownValueError:
print("Could not understand audio")
except sr.RequestError as e:
print("Could not request results. check your internet connection")
i += 1
os.chdir('..')
if __name__ == '__main__':
print('Enter the audio file path')
path = input()
2
silence_based_conversion(path)
```