// const player = require('play-sound')(opts = {player: "/path/to/downloads/mplayer/mplayer.exe"})

// const play = require('audio-play');

// const load = require('audio-loader');

// const AudioRecorder = require('node-audiorecorder');

const fs = require('fs');

const path = require('path');

const DIRECTORY = `examples-recordings`;

const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

let answers = [];

let timeIsUp = false;

const player = require('node-wav-player');

function randomInt(low, high) {

return Math.floor(Math.random() \* (high - low) + low)

}

function playTimes() {

return new Promise(resolve => {

player.play({

path: 'times.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function playNumber(number) {

return new Promise(resolve => {

player.play({

path: number + '.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function inputNumber() {

return new Promise(resolve => {

rl.question('The answer? ', (answer) => {

// TODO: Log the answer in a database

console.log(`Thank you for your valuable feedback: ${answer}`);

resolve(answer)

// rl.close();

});

});

}

function playEquals() {

return new Promise(resolve => {

player.play({

path: 'equals.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function playIncorrect() {

return new Promise(resolve => {

player.play({

path: 'incorrect.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function playCorrect() {

return new Promise(resolve => {

player.play({

path: 'correct.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function playThanks() {

return new Promise(resolve => {

player.play({

path: 'ThankYou.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function playLeaveRoom() {

return new Promise(resolve => {

player.play({

path: 'LeaveRoom.wav',

sync: true

}).then(() => {

resolve();

// console.log('The wav file was played through.');

}).catch((error) => {

console.error(error);

});

})

}

function record() {

return new Promise(resolve => {

// Options is an optional parameter for the constructor call.

// If an option is not given the default value, as seen below, will be used.

const options = {

program: `rec`, // Which program to use, either `arecord`, `rec`, or `sox`.

device: null, // Recording device to use.

bits: 16, // Sample size. (only for `rec` and `sox`)

channels: 1, // Channel count.

encoding: `signed-integer`, // Encoding type. (only for `rec` and `sox`)

format: `S16\_LE`, // Encoding type. (only for `arecord`)

rate: 16000, // Sample rate.

type: `wav`, // Format type.

// Following options only available when using `rec` or `sox`.

silence: 2, // Duration of silence in seconds before it stops recording.

thresholdStart: 0, // Silence threshold to start recording.

thresholdStop: 0.25, // Silence threshold to stop recording.

keepSilence: true // Keep the silence in the recording.

};

// Optional parameter intended for debugging.

// The object has to implement a log and warn function.

const logger = console;

// Create an instance.

let audioRecorder = new AudioRecorder(options, logger);

if (!fs.existsSync(DIRECTORY)) {

fs.mkdirSync(DIRECTORY);

}

const fileName = path.join(DIRECTORY, Math.random().toString(36).replace(/[^a-z]+/g, ``).substr(0, 4).concat(`.wav`));

console.log(`Writing new recording file at: `, fileName);

// Create write stream.

const fileStream = fs.createWriteStream(fileName, { encoding: `binary` });

// Start and write to the file.

audioRecorder.start().stream().pipe(fileStream);

// Log information on the following events

audioRecorder.stream().on(`close`, function (code) {

resolve(fileName);

console.warn(`Recording closed. Exit code: `, code);

});

});

}

async function speechToText(recordedFile) {

// Imports the Google Cloud client library

const speech = require('@google-cloud/speech');

const fs = require('fs');

// Creates a client

const client = new speech.SpeechClient();

// The name of the audio file to transcribe

const fileName = '=.m4a';

// Reads a local audio file and converts it to base64

const file = fs.readFileSync(recordedFile);

const audioBytes = file.toString('base64');

// The audio file's encoding, sample rate in hertz, and BCP-47 language code

const audio = {

content: audioBytes,

};

const config = {

encoding: 'LINEAR16',

sampleRateHertz: 16000,

// languageCode: 'yue',

languageCode: 'en-US',

};

const request = {

audio: audio,

config: config,

};

// Detects speech in the audio file

return await client.recognize(request);

}

function checkAnswer(number1, number2, userAnswer) {

return new Promise(resolve => {

let answer = number1 \* number2;

console.log('answer: ' + answer);

if (userAnswer == answer) {

resolve(true)

} else {

resolve(false)

}

})

}

function correctanswers(){

let answersGood = 0;

let answersWrong = 0;

for (let answer of answers) {

if (answer) {

answersGood = answersGood + 1;

} else {

answersWrong = answersWrong + 1;

}

}

console.log(`Answers correct: ${answersGood}/${answers.length}`);

}

function loop () {

let number1 = randomInt(1, 10);

let number2 = randomInt(1, 100);

console.log(number1 + ' \* ' + number2);

if (!timeIsUp) {

return playNumber(number1)

.then(() => playTimes())

.then(() => playNumber(number2))

.then(() => inputNumber())

.then(answer => {

return checkAnswer(number1, number2, answer);

})

.then(resultIsCorrect => {

answers.push(resultIsCorrect);

if (resultIsCorrect) {

return playCorrect()

} else {

return playIncorrect()

}

})

.then(loop)

.catch(err => {

console.log(err);

})

} else {

let answersGood = 0;

let answersWrong = 0;

for (let answer of answers) {

if (answer) {

answersGood = answersGood + 1;

} else {

answersWrong = answersWrong + 1;

}

}

console.log(`Answers correct: ${answersGood}/${answers.length}`);

return playThanks()

.then(() => playNumber(answersGood))

.then(() => playLeaveRoom())

process.exit(1);

}

}

setTimeout(function(){

timeIsUp = true;

}, 300000 )

loop();