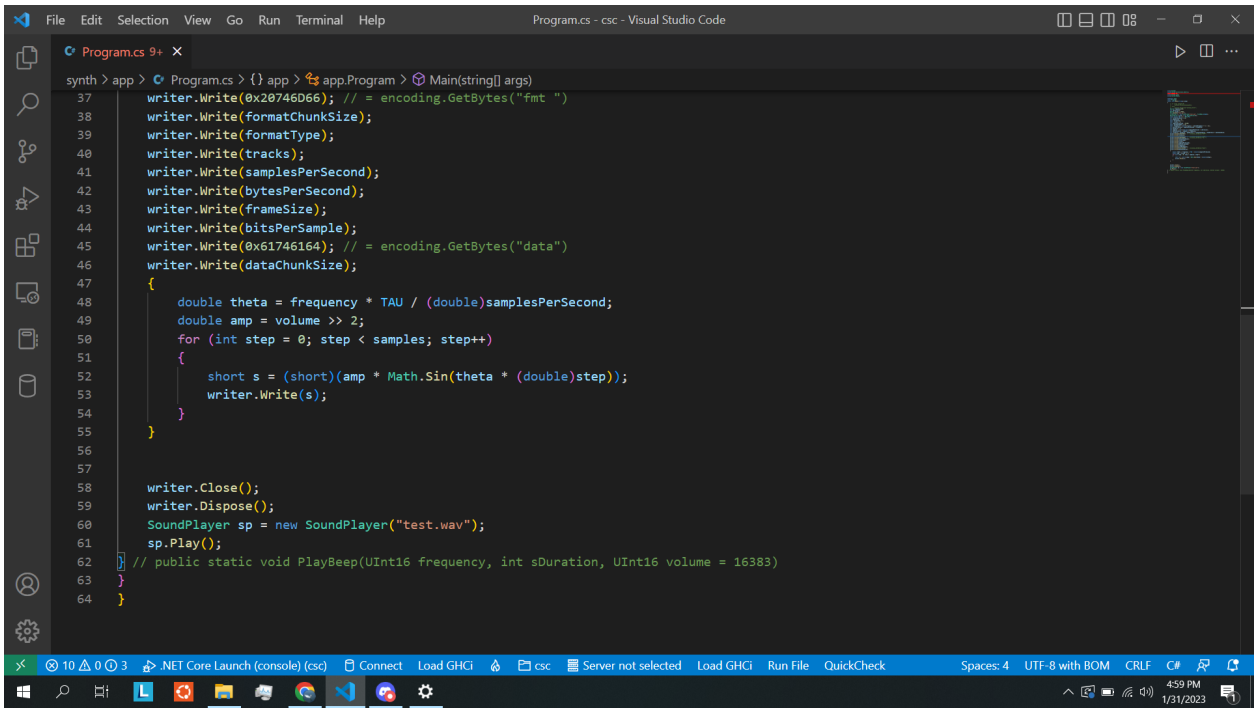
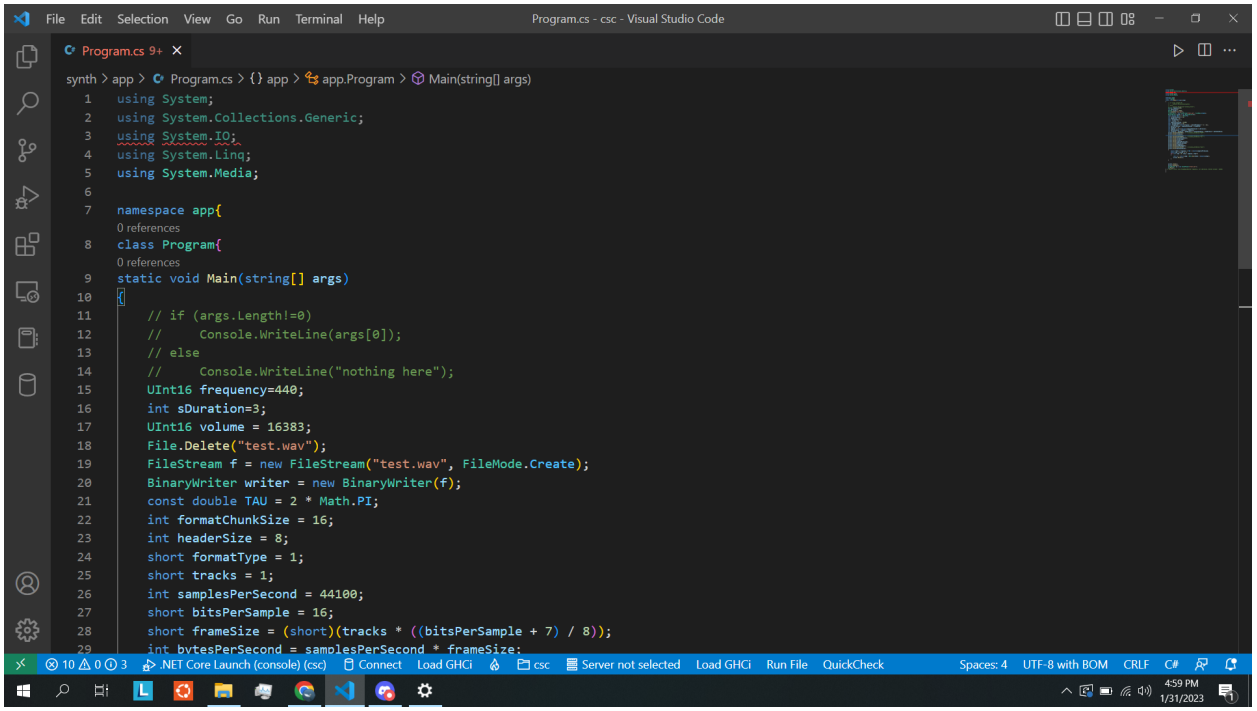
Skylar Buck

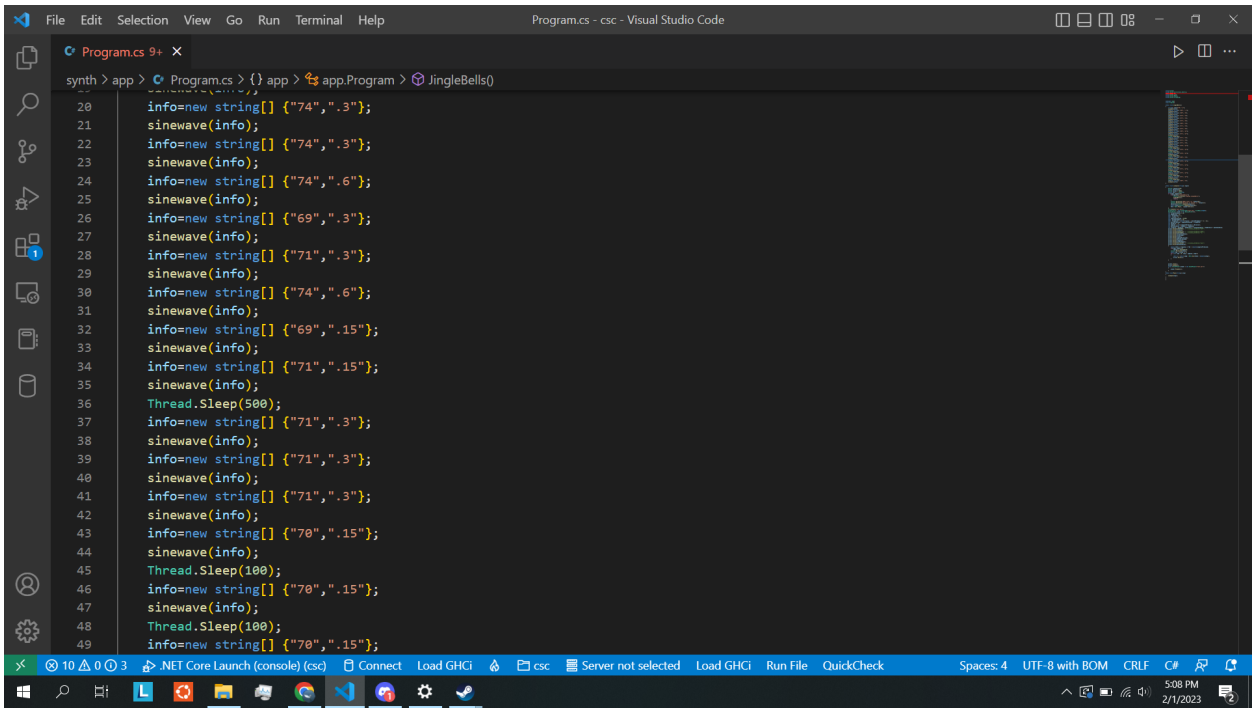
Each question is separated by screenshots. We start with question 1.

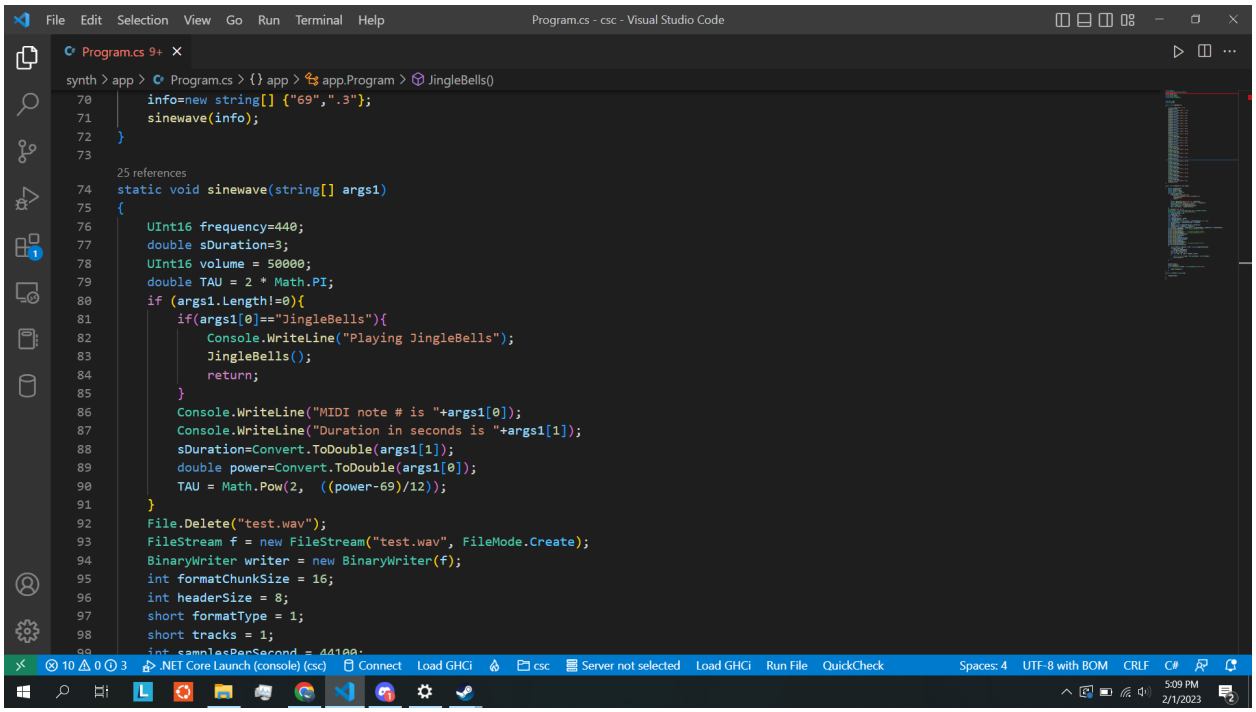




For question 1 we had to create audio for 3 seconds of a 440Hz sinusoid with a sampling rate of 44100 samples per second. The first thing i had to do was create a wav file so that I could play the sound from it. In order to do this in c# you have to write each individual thing. This includes the file size, file type, format, how many tracks there are, samples per second, bits per sample, ect.. I named this file test.wav and once all the writing to the file to make the skeleton is

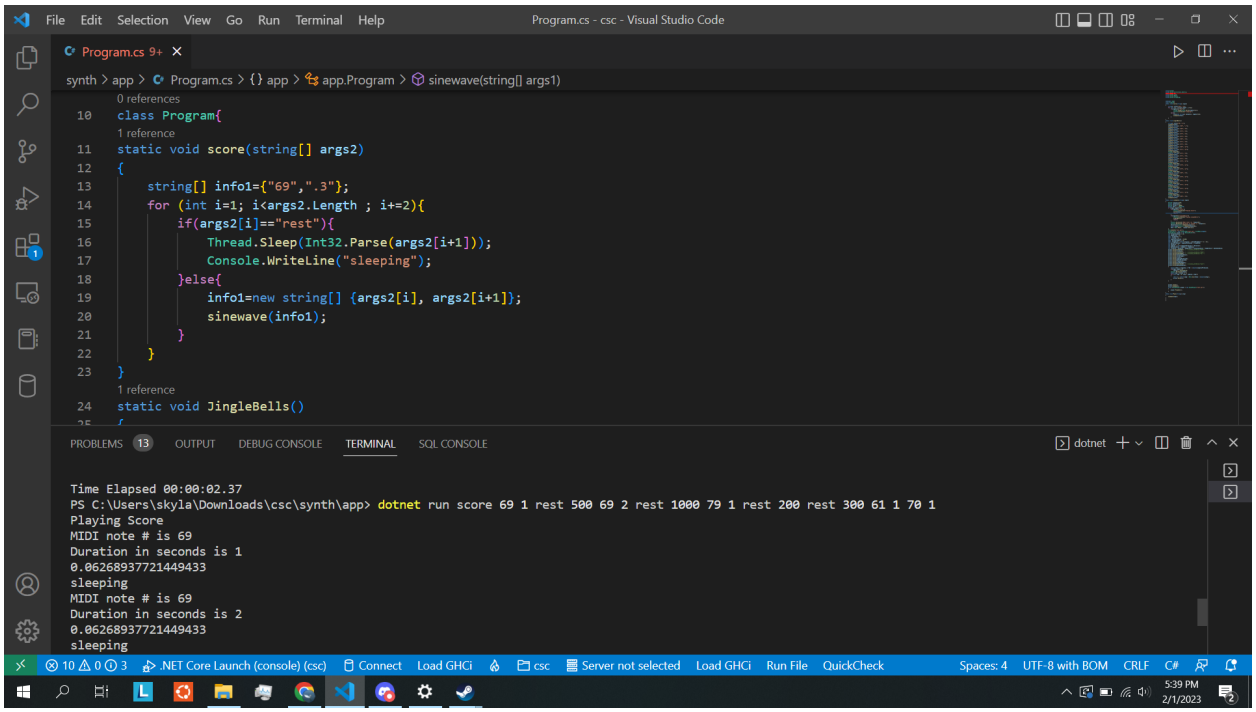
complete we use formulas to compute theta and amp. Then using those we generate shorts that are written to the file. Each short is calculated using theta, amp, math.sin, and step(from 0 to number of samples). Once all of this is written to the file we can use SoundPlayer to run the audio for the file.



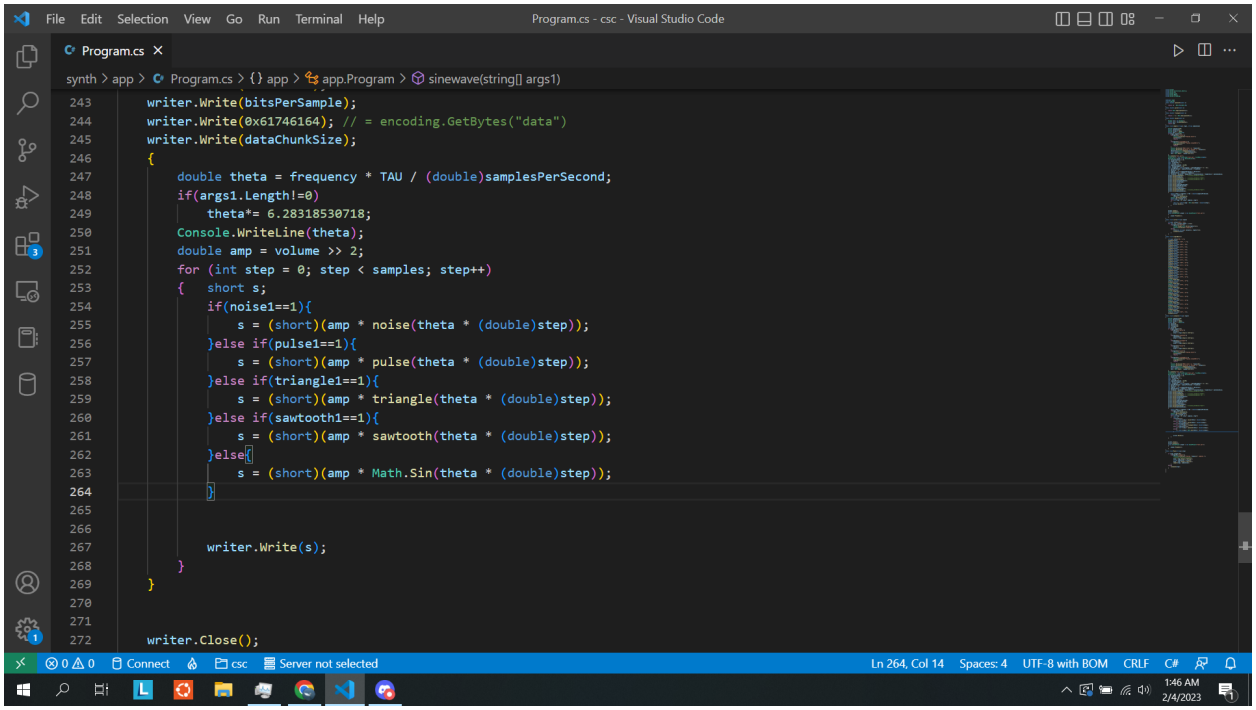


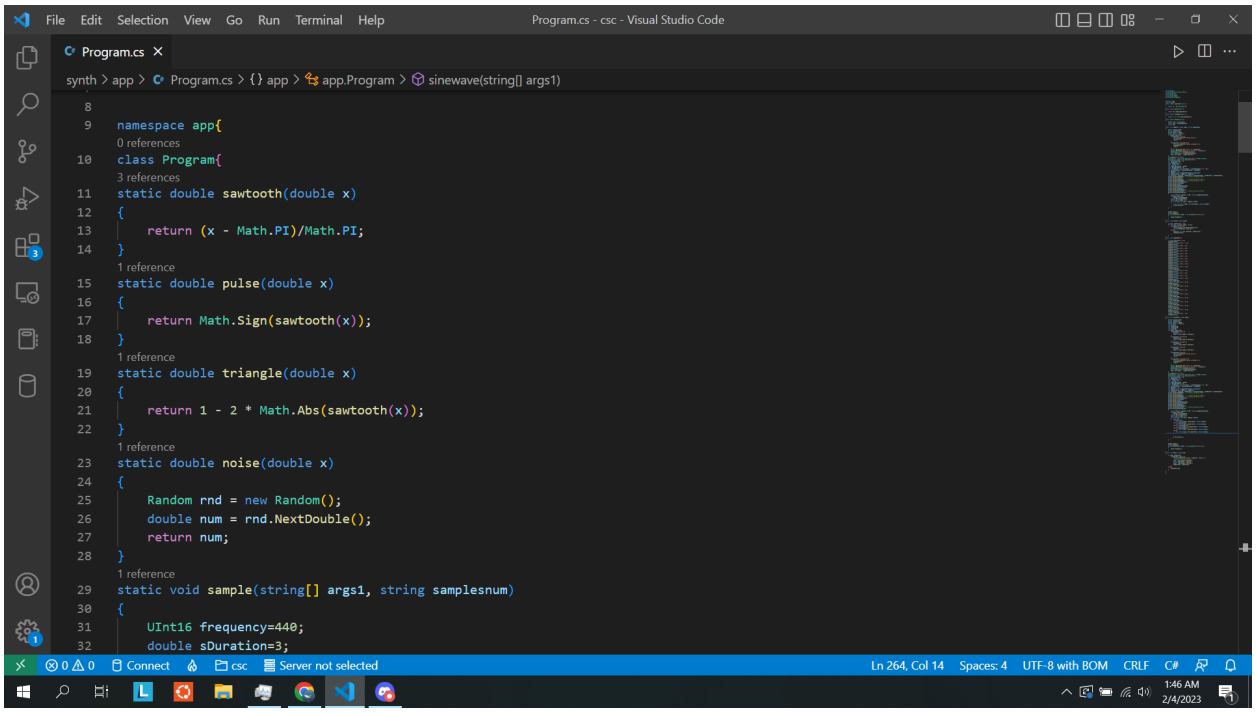
For question 2 we had to create a function that takes as arguments a MIDI note number and a duration and similarly generates a sinusoidal sound. We also had to make a song. For the song

I chose to do the famous christmas carol “jingle bells” which is the function shown in the first screenshot. In the second screenshot we are using the command line inputs to either run the jingle bells song or alter frequency/duration based on what is put into the arguments. The formula is used to calculate TAU by changing the midi note into something we can use in order to generate the correct sound.

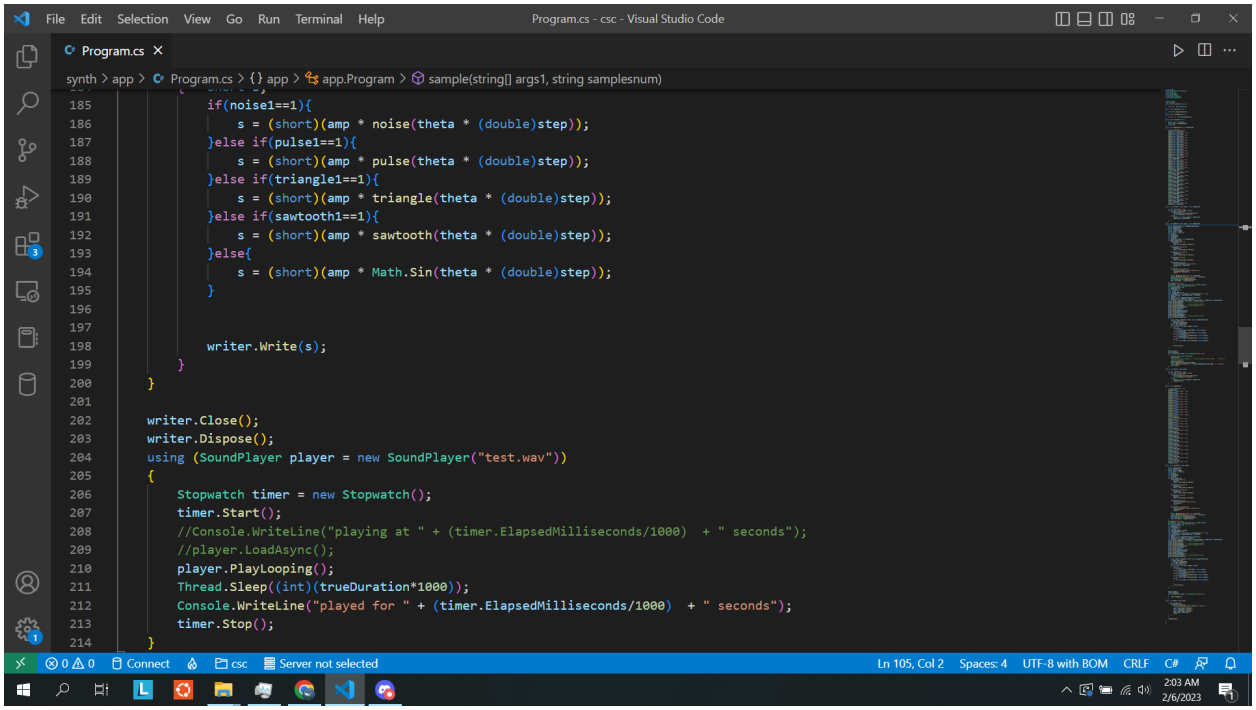


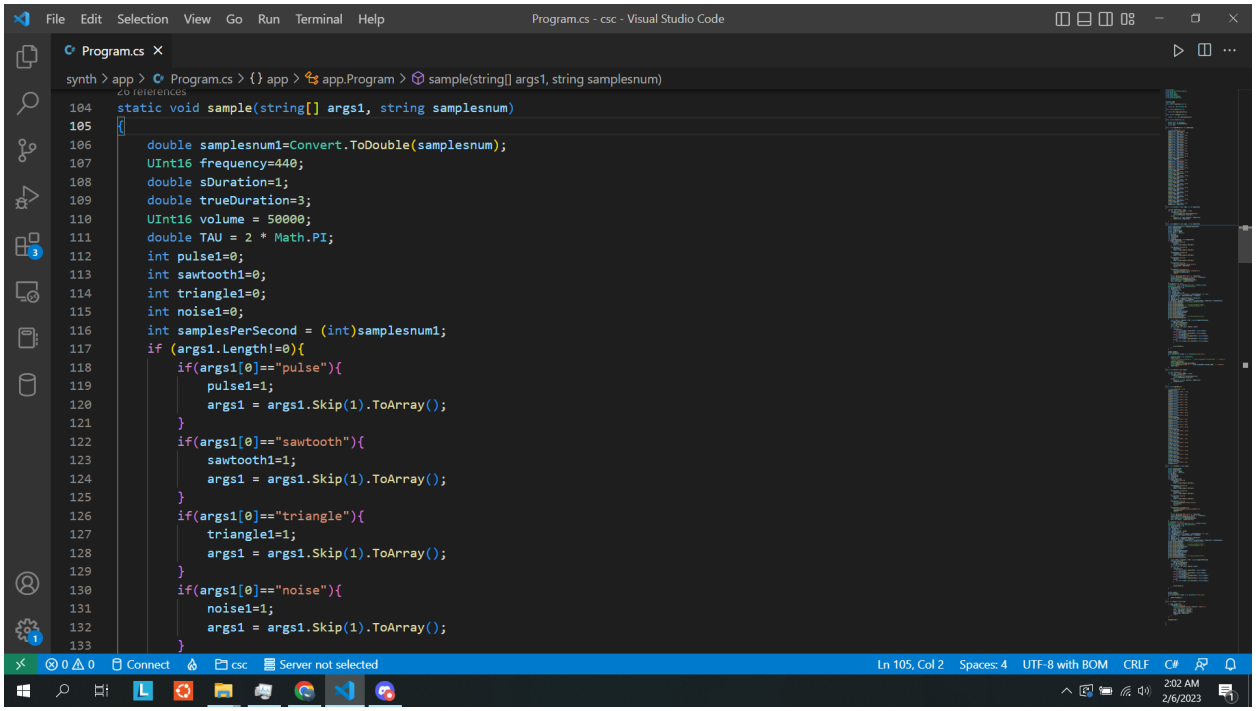
For question 3 we can now run multiple notes in succession and include rests as well. For command line input it reads score, goes to this function and skips the first input(“score”). The loop that takes the command line as input will take the first input and will register that as either a note or as a rest with the second input as amount of time in seconds. Rest will sleep for the amount of seconds allotted while the note will play for the amount of seconds allotted. Then the loop will reset and look at the next two inputs until there are no more inputs to look at.

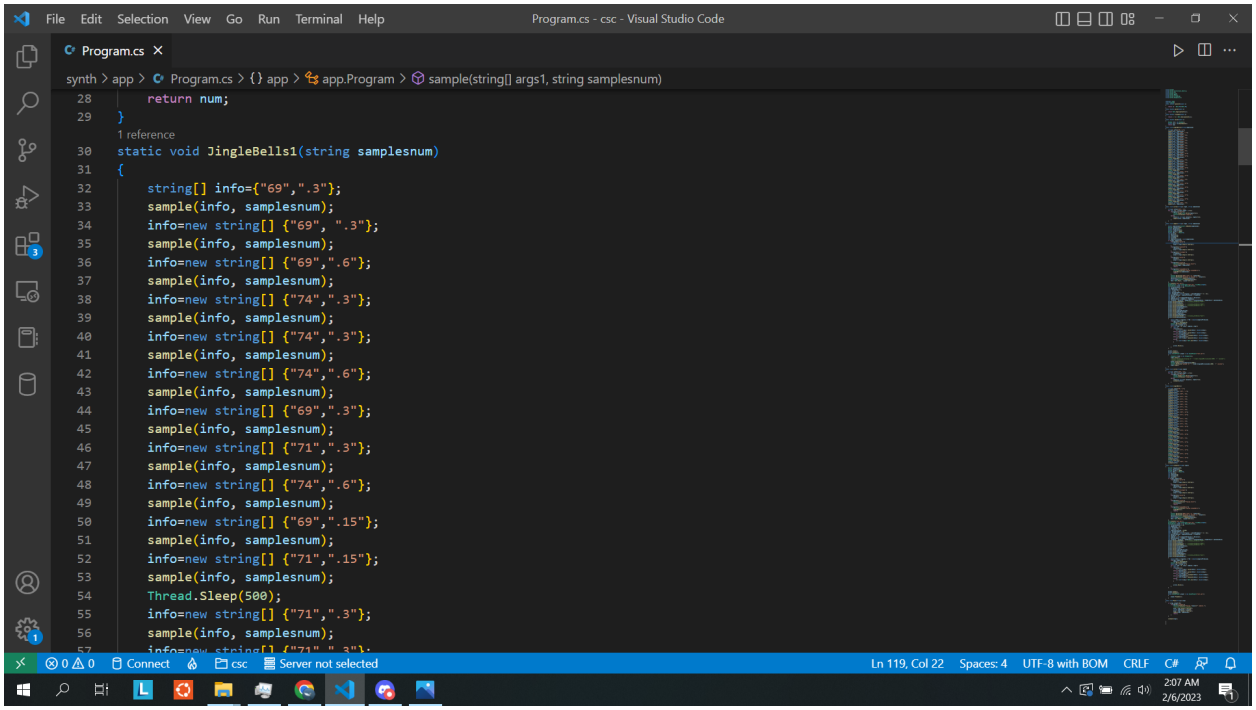


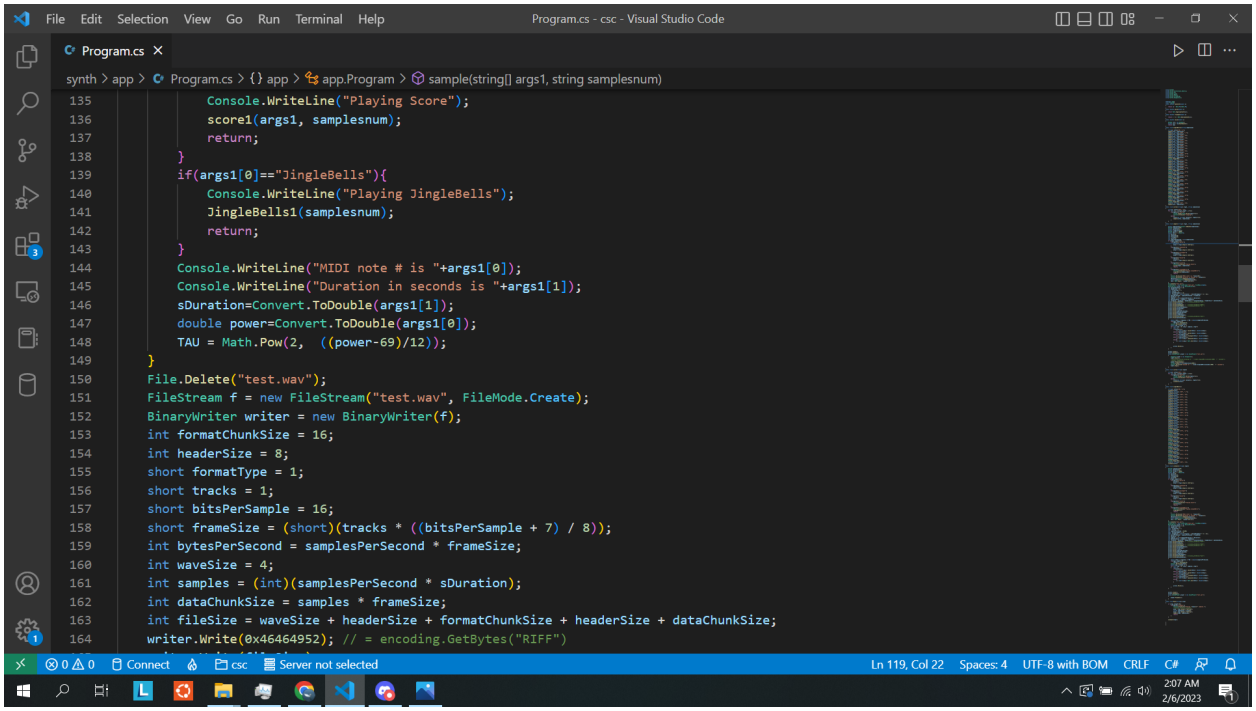


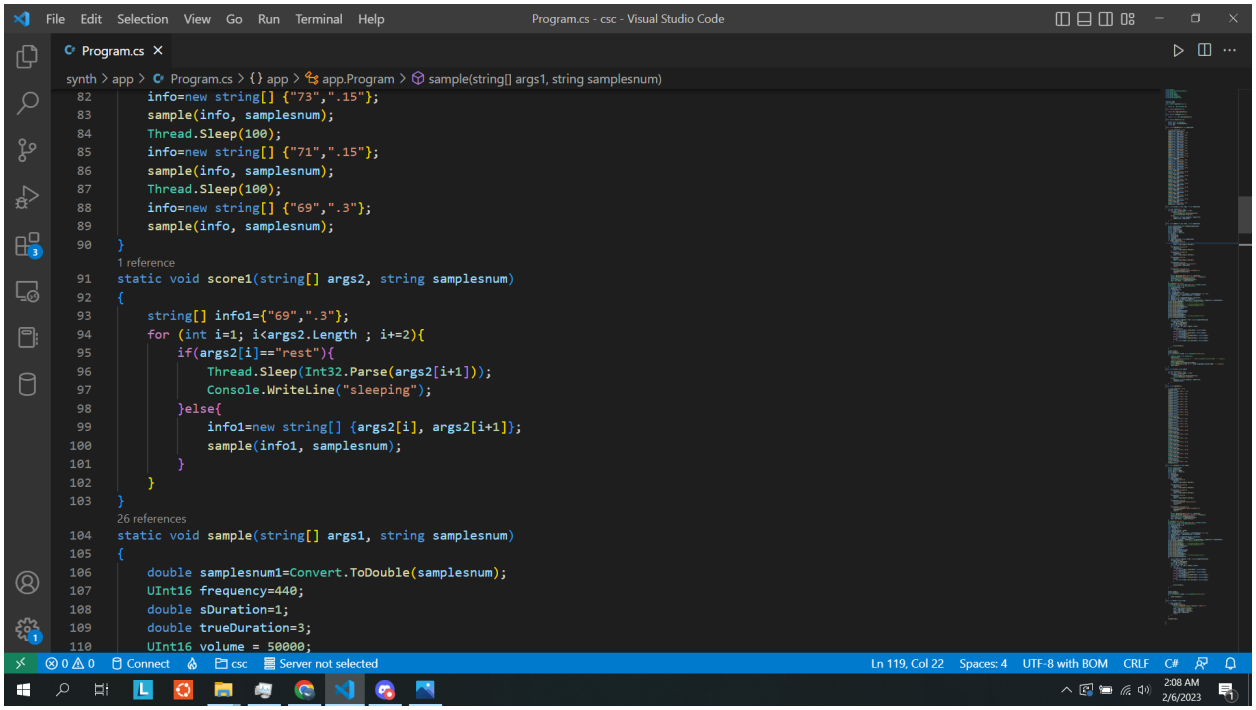
For question 4 we had to Implement the following oscillators: Sawtooh, Pulse, Triangle and Noise. The 2nd screenshot shows the formulas for each of these oscillators as their own function that gets called in the step loop that gets written into the file. This step loop is shown in the first screenshot, the oscillator that gets chosen depends on what is determined in the command line arguments which declares which one to use and causes the variables to get ticked.

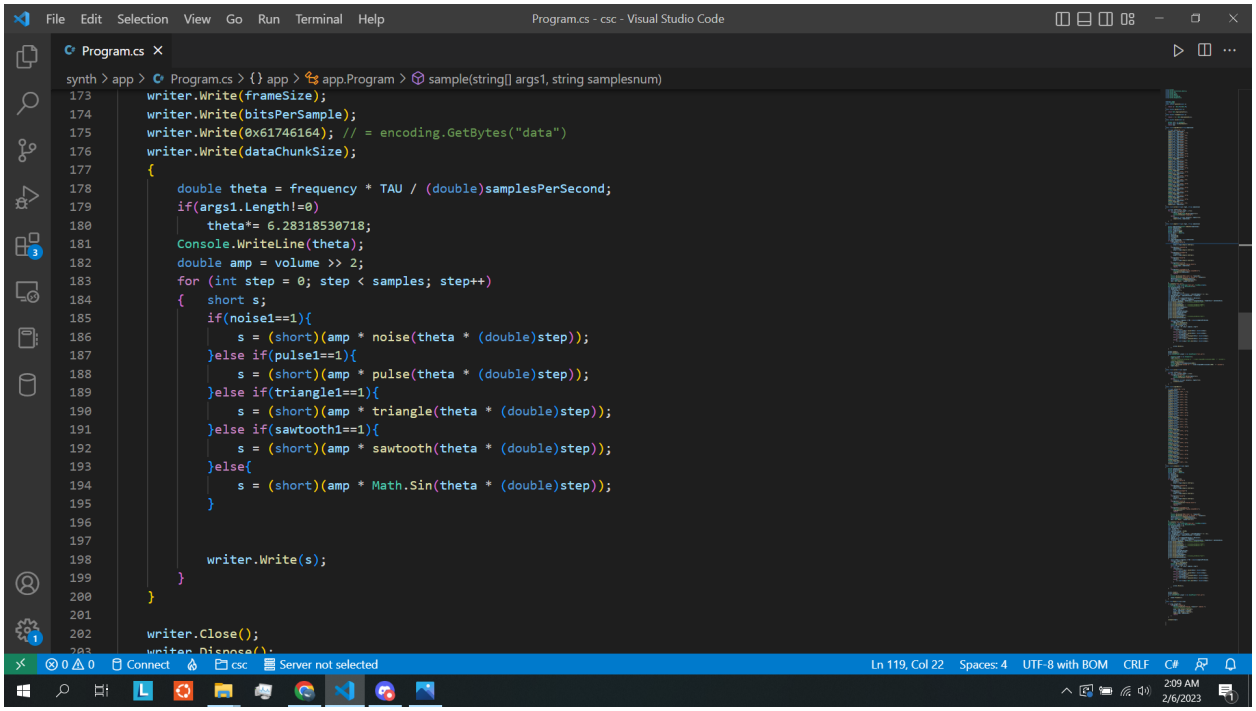

File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
if(noise1==1){
s = (short)(amp * noise(theta * (double)step));
}else if(pulse1==1){
s = (short)(amp * pulse(theta * (double)step));
}else if(triangle1==1){
s = (short)(amp * triangle(theta * (double)step));
}else if(sawtooth1==1){
s = (short)(amp * sawtooth(theta * (double)step));
}else{
s = (short)(amp * Math.Sin(theta * (double)step));
}
writer.Write(s);
writer.Close();
writer.Dispose();
using (SoundPlayer player = new SoundPlayer("test.wav"))
{
Stopwatch timer = new Stopwatch();
timer.Start();
//Console.WriteLine("playing at " + (timer.ElapsedMilliseconds/1000) + " seconds");
//player.LoadAsync();
player.PlayLooping();
Thread.Sleep((int)(trueDuration*1000));
Console.WriteLine("played for " + (timer.ElapsedMilliseconds/1000) + " seconds");
timer.Stop();
}
Connect
csc
Server not selected
Ln 105, Col 2
Spaces: 4
UTF-8 with BOM
CRLF
C#
2:03 AM
2/6/2023

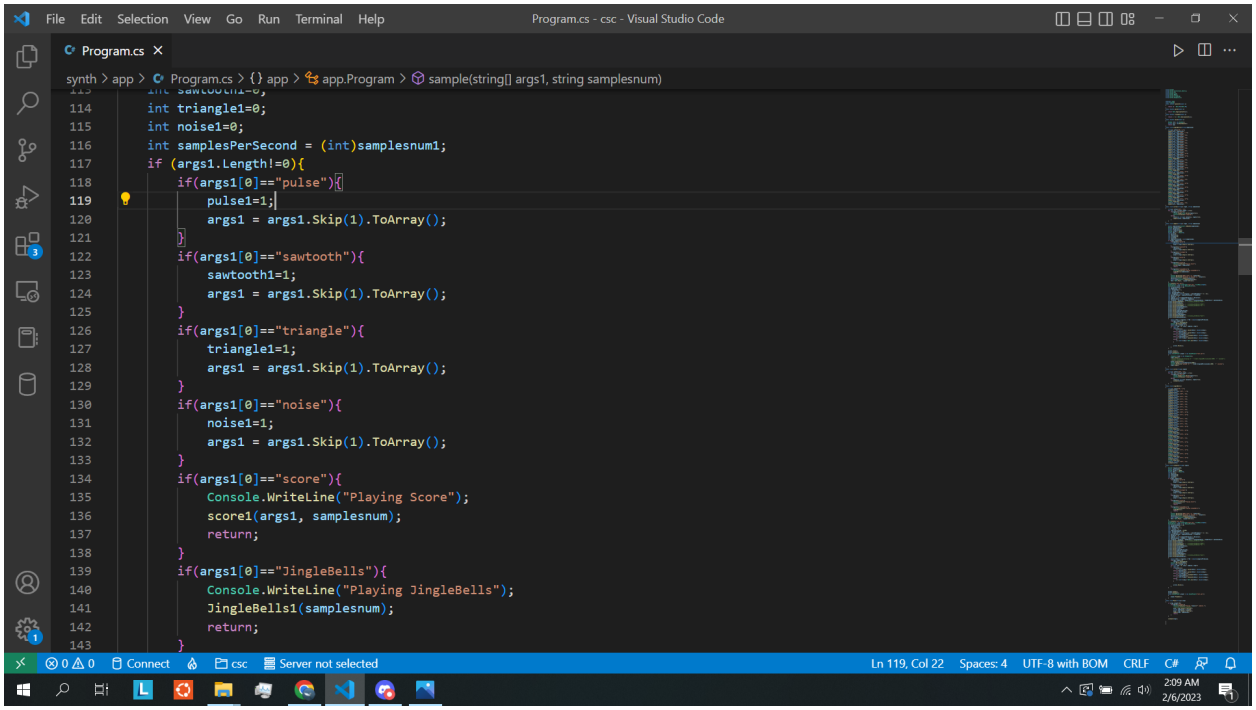

File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
20 references
static void sample(string[] args1, string samplesnum)
{
double samplesnum1=Convert.ToDouble(samplesnum);
UInt16 frequency=440;
double sDuration=1;
double trueDuration=3;
UInt16 volume = 50000;
double TAU = 2 * Math.PI;
int pulse1=0;
int sawtooth1=0;
int triangle1=0;
int noise1=0;
int samplesPerSecond = (int)samplesnum1;
if (args1.Length!=0){
if(args1[0]=="pulse"){
pulse1=1;
args1 = args1.Skip(1).ToArray();
}
if(args1[0]=="sawtooth"){
sawtooth1=1;
args1 = args1.Skip(1).ToArray();
}
if(args1[0]=="triangle"){
triangle1=1;
args1 = args1.Skip(1).ToArray();
}
if(args1[0]=="noise"){
noise1=1;
args1 = args1.Skip(1).ToArray();
}
Connect
csc
Server not selected
Ln 105, Col 2
Spaces: 4
UTF-8 with BOM
CRLF
C#
2:02 AM
2/6/2023

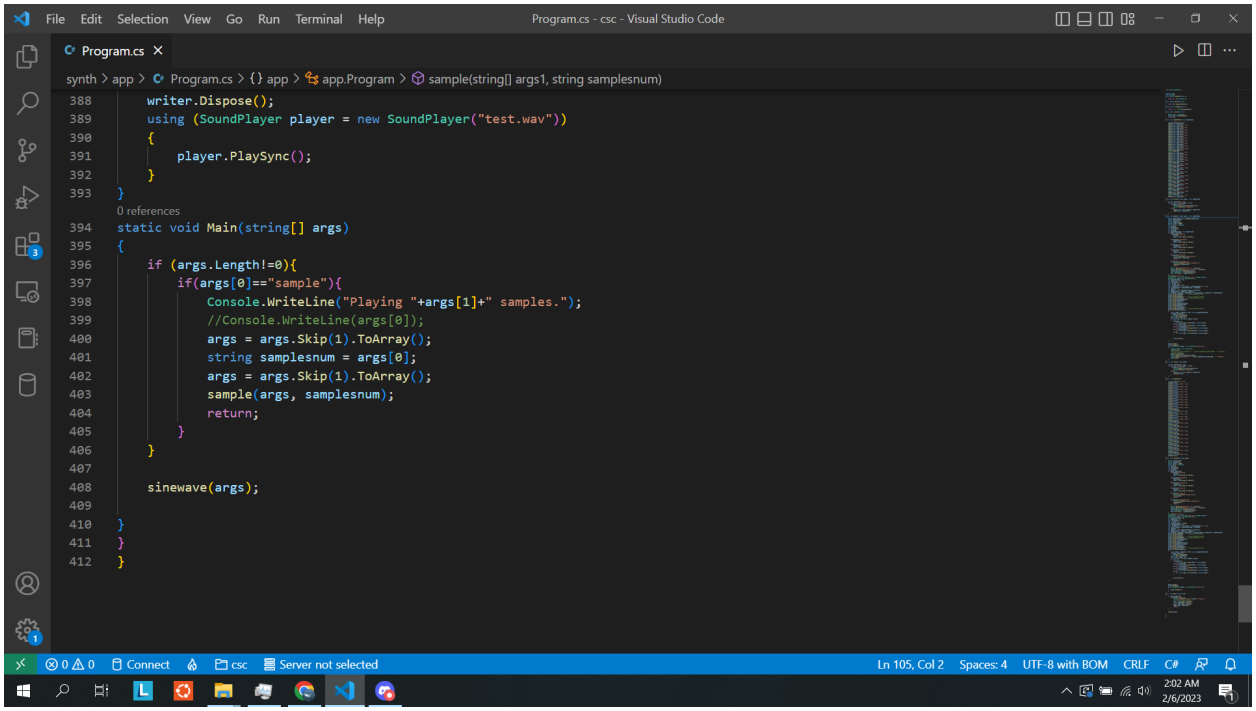

File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
return num;
}
1 reference
static void JingleBells1(string samplesnum)
{
string[] info={"69",".3"};
sample(info, samplesnum);
info=new string[] {"69", ".3"};
sample(info, samplesnum);
info=new string[] {"69",".6"};
sample(info, samplesnum);
info=new string[] {"74",".3"};
sample(info, samplesnum);
info=new string[] {"74",".3"};
sample(info, samplesnum);
info=new string[] {"74",".6"};
sample(info, samplesnum);
info=new string[] {"69",".3"};
sample(info, samplesnum);
info=new string[] {"71",".3"};
sample(info, samplesnum);
info=new string[] {"74",".6"};
sample(info, samplesnum);
info=new string[] {"69",".15"};
sample(info, samplesnum);
info=new string[] {"71",".15"};
sample(info, samplesnum);
Thread.Sleep(500);
info=new string[] {"71",".3"};
sample(info, samplesnum);
0 0 Connect csc Server not selected
Ln 119, Col 22 Spaces: 4 UTF-8 with BOM CRLF C#
2:07 AM
2/6/2023


File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
Console.WriteLine("Playing Score");
score1(args1, samplesnum);
return;
}
if(args1[0]=="JingleBells"){
Console.WriteLine("Playing JingleBells");
JingleBells1(samplesnum);
return;
}
Console.WriteLine("MIDI note # is "+args1[0]);
Console.WriteLine("Duration in seconds is "+args1[1]);
sDuration=Convert.ToDouble(args1[1]);
double power=Convert.ToDouble(args1[0]);
TAU = Math.Pow(2, ((power-69)/12));
}
File.Delete("test.wav");
FileStream f = new FileStream("test.wav", FileMode.Create);
BinaryWriter writer = new BinaryWriter(f);
int formatChunkSize = 16;
int headerSize = 8;
short formatType = 1;
short tracks = 1;
short bitsPerSample = 16;
short frameSize = (short)(tracks * ((bitsPerSample + 7) / 8));
int bytesPerSecond = samplesPerSecond * frameSize;
int waveSize = 4;
int samples = (int)(samplesPerSecond * sDuration);
int dataChunkSize = samples * frameSize;
int fileSize = waveSize + headerSize + formatChunkSize + headerSize + dataChunkSize;
writer.Write(0x46464952); // = encoding.GetBytes("RIFF")
0 0 Connect csc Server not selected
Ln 119, Col 22 Spaces: 4 UTF-8 with BOM CRLF C#
2:07 AM
2/6/2023


File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
info=new string[] {"73",".15"};
sample(info, samplesnum);
Thread.Sleep(100);
info=new string[] {"71",".15"};
sample(info, samplesnum);
Thread.Sleep(100);
info=new string[] {"69",".3"};
sample(info, samplesnum);
}
1 reference
static void score1(string[] args2, string samplesnum)
{
string[] info1={"69",".3"};
for (int i=1; i<args2.Length ; i+=2){
if(args2[i]=="rest"){
Thread.Sleep(Int32.Parse(args2[i+1]));
Console.WriteLine("sleeping");
}else{
info1=new string[] {args2[i], args2[i+1]};
sample(info1, samplesnum);
}
}
}
26 references
static void sample(string[] args1, string samplesnum)
{
double samplesnum1=Convert.ToDouble(samplesnum);
UInt16 frequency=440;
double sDuration=1;
double trueDuration=3;
UInt16 volume = 50000;
0 0 Connect csc Server not selected
Ln 119, Col 22 Spaces: 4 UTF-8 with BOM CRLF C#
2:08 AM
2/6/2023

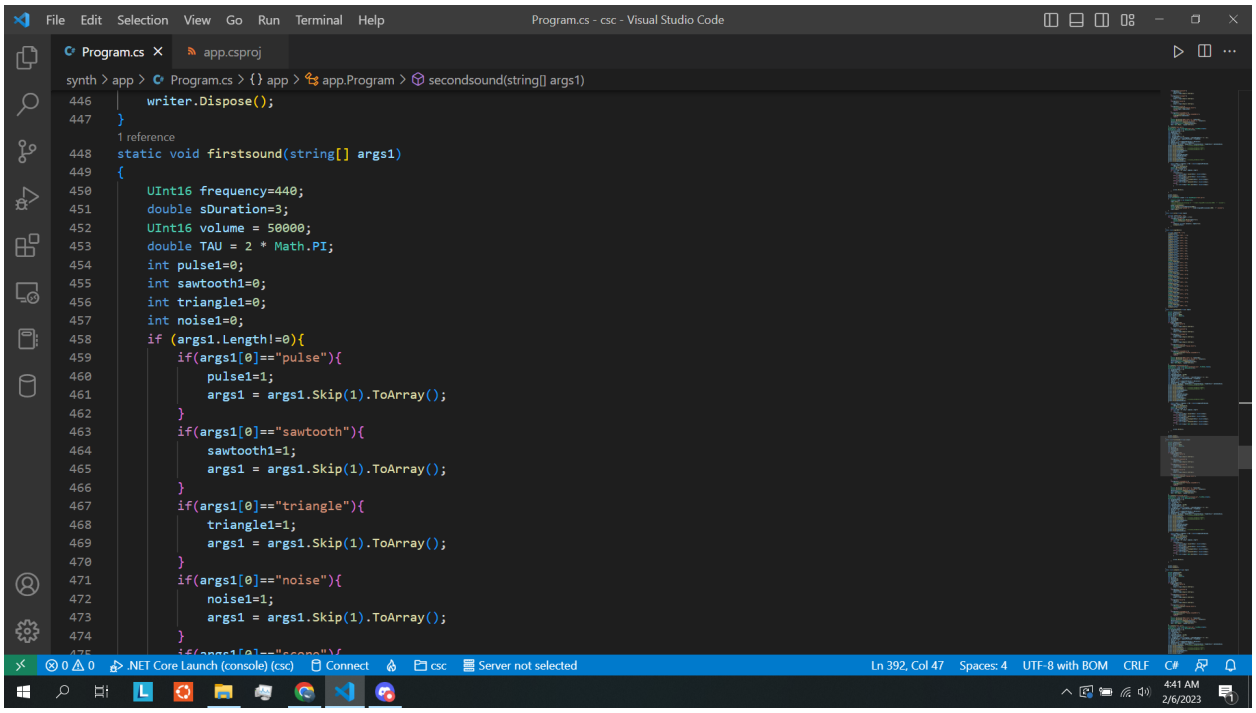

File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
synth > app > Program.cs > { } app > app.Program > sample(string[] args1, string samplesnum)
writer.Write(frameSize);
writer.Write(bitsPerSample);
writer.Write(0x61746164); // = encoding.GetBytes("data")
writer.Write(dataChunkSize);
{
double theta = frequency * TAU / (double)samplesPerSecond;
if(args1.Length!=0)
theta*= 6.28318530718;
Console.WriteLine(theta);
double amp = volume >> 2;
for (int step = 0; step < samples; step++)
{ short s;
if(noise1==1){
s = (short)(amp * noise(theta * (double)step));
}else if(pulse1==1){
s = (short)(amp * pulse(theta * (double)step));
}else if(triangle1==1){
s = (short)(amp * triangle(theta * (double)step));
}else if(sawtooth1==1){
s = (short)(amp * sawtooth(theta * (double)step));
}else{
s = (short)(amp * Math.Sin(theta * (double)step));
}
writer.Write(s);
}
}
writer.Close();
0 0 Connect csc Server not selected
Ln 119, Col 22 Spaces: 4 UTF-8 with BOM CRLF C#
2:09 AM
2/6/2023

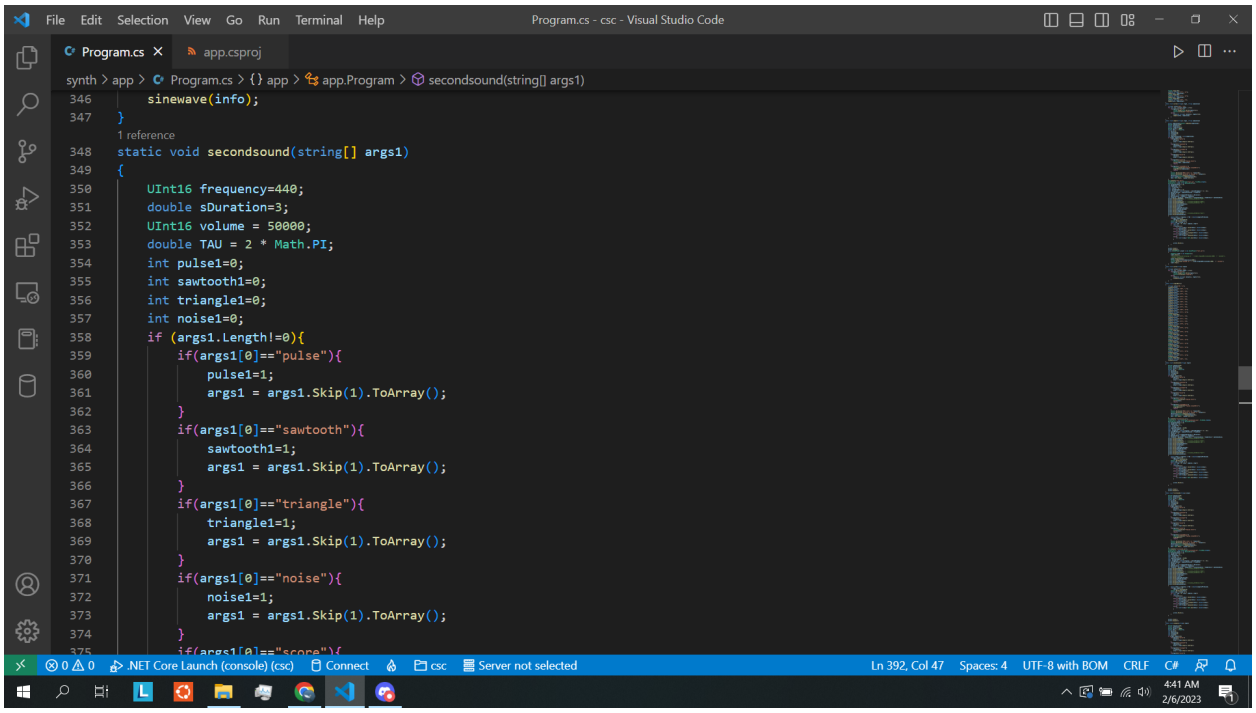


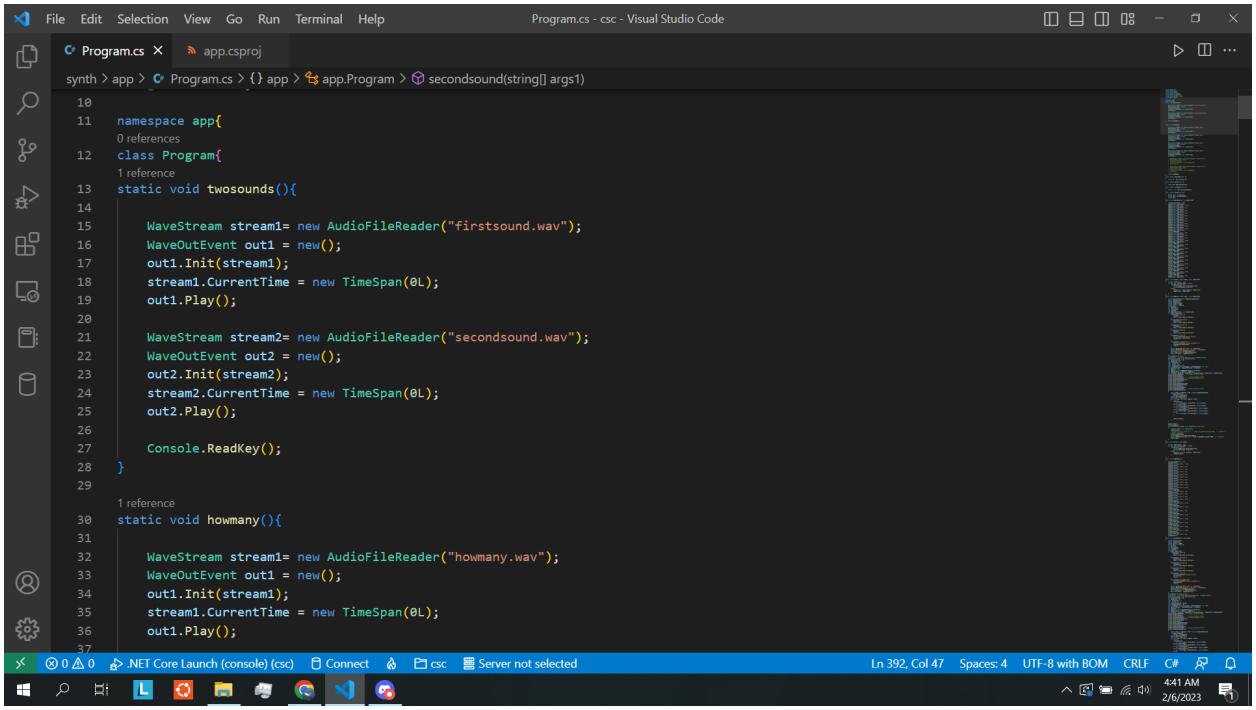


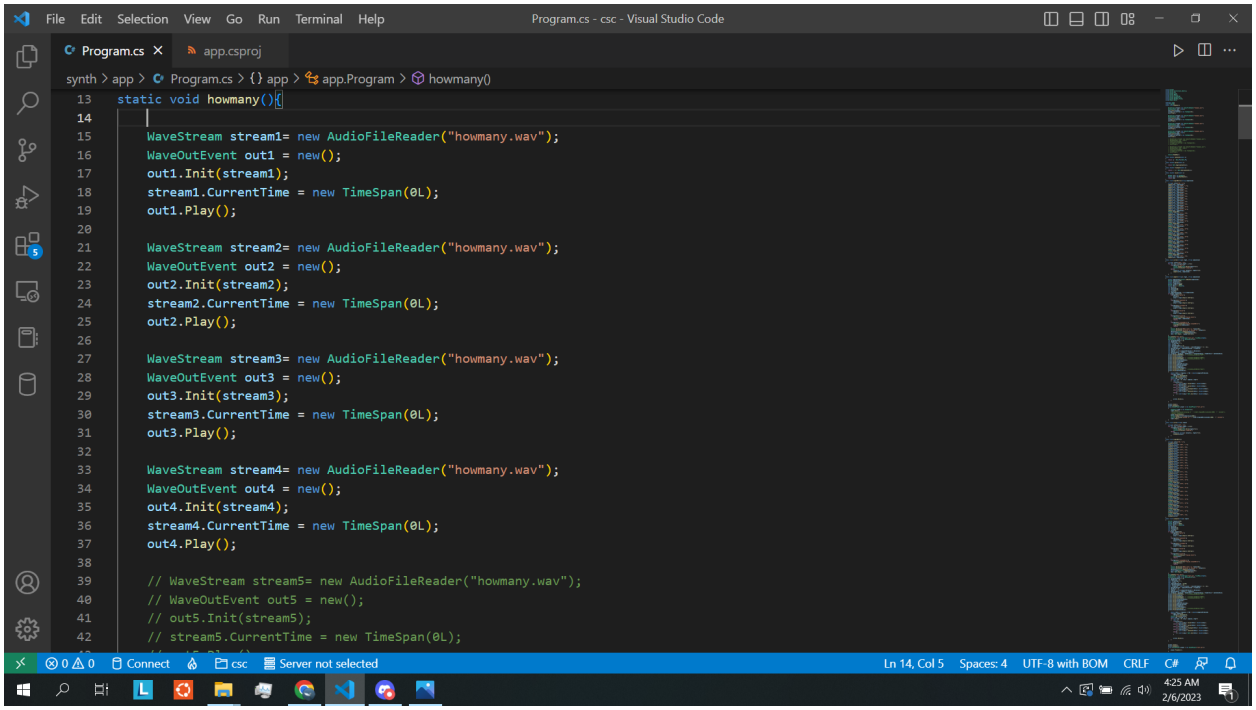
For question 5 we had to re-implement questions 1,2,3,4 but using buffers. I decided to create a sinusoid for 1 second then my oscillator function will be called multiple times to generate a user-provided number of samples. To do this i essentially copied the whole file and altered it to make only a 1 second buffer of sound then I would replay the sound for the amount of time that is needed, even if the sound is need to be 0.5 seconds or 2.5 seconds of duration. For command line arguments by default it takes 2 which is the word sample followed by how many samples they would like to play. Any commend line

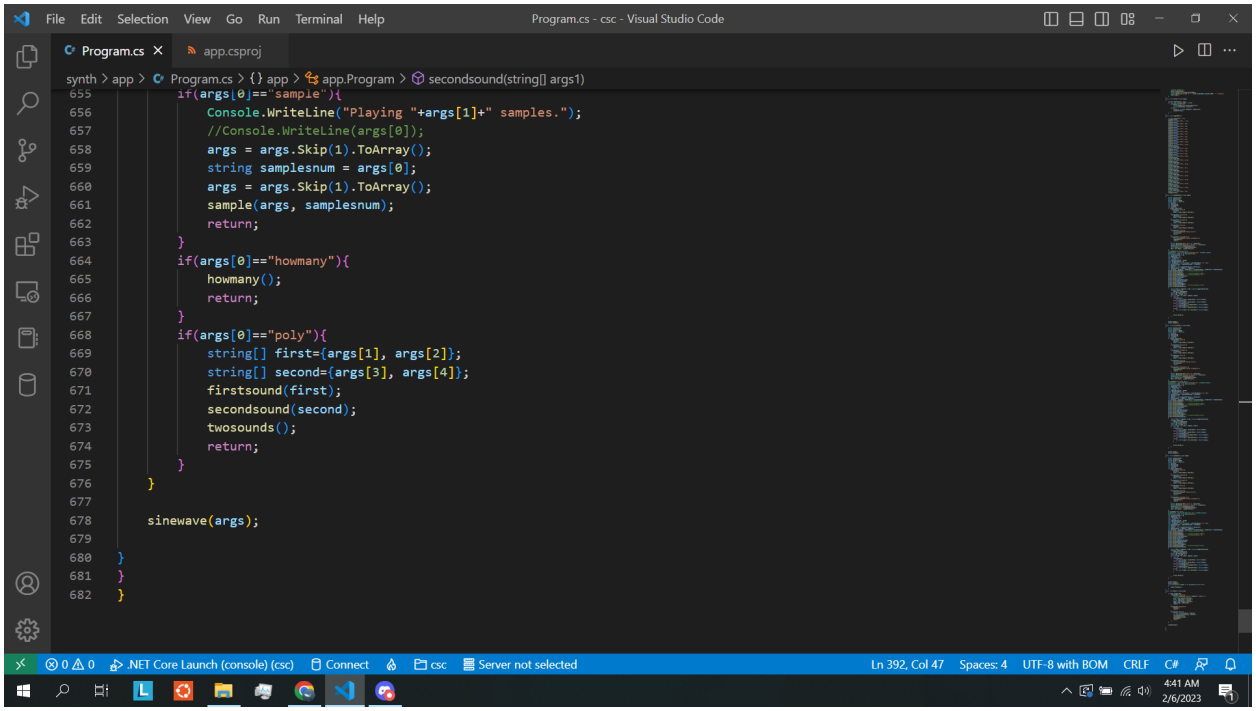
arguments past that go into the questions as MIDI note number and a duration as well as jingle bells for question 2, score(string of many notes and rests) for question 3, and different oscillators for question 4. Question 1 is re-implemented in screenshot 8, screenshot 1, screenshot 2, and screenshot 4. Question 2 is re-implemented in screenshot 3 and screenshot 4. Question 3 is re-implemented in screenshot 4 and screenshot 5. Question 4 is re-implemented in screenshot 7, screenshot 6, and screenshot 2.





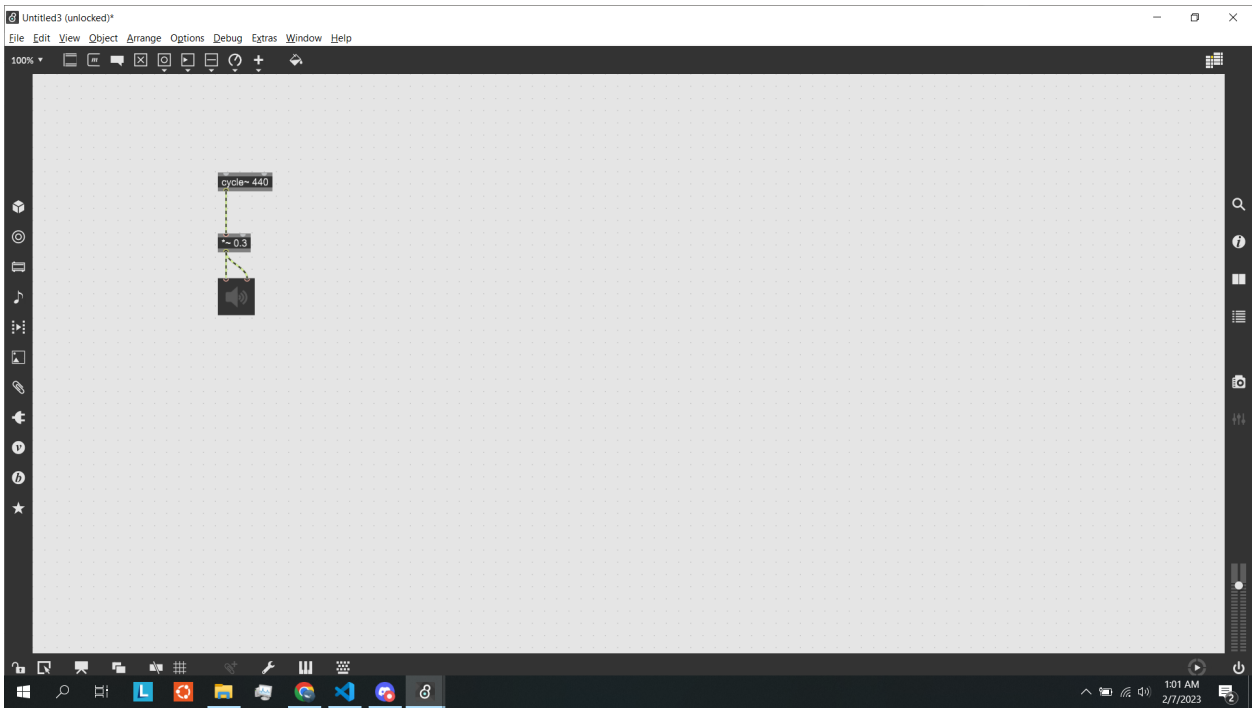

File Edit Selection View Go Run Terminal Help
Program.cs - csc - Visual Studio Code
Program.cs
app.csproj
synth > app > Program.cs > { } app > app.Program > secondsound(string[] args1)
namespace app{
0 references
class Program{
1 reference
static void twosounds(){
WaveStream stream1= new AudioFileReader("firstsound.wav");
WaveOutEvent out1 = new();
out1.Init(stream1);
stream1.CurrentTime = new TimeSpan(0L);
out1.Play();
WaveStream stream2= new AudioFileReader("secondsound.wav");
WaveOutEvent out2 = new();
out2.Init(stream2);
stream2.CurrentTime = new TimeSpan(0L);
out2.Play();
Console.ReadKey();
}
1 reference
static void howmany(){
WaveStream stream1= new AudioFileReader("howmany.wav");
WaveOutEvent out1 = new();
out1.Init(stream1);
stream1.CurrentTime = new TimeSpan(0L);
out1.Play();
.NET Core Launch (console) (csc)
Connect
csc
Server not selected
Ln 392, Col 47
Spaces: 4
UTF-8 with BOM
CRLF
C#
4:41 AM
2/6/2023

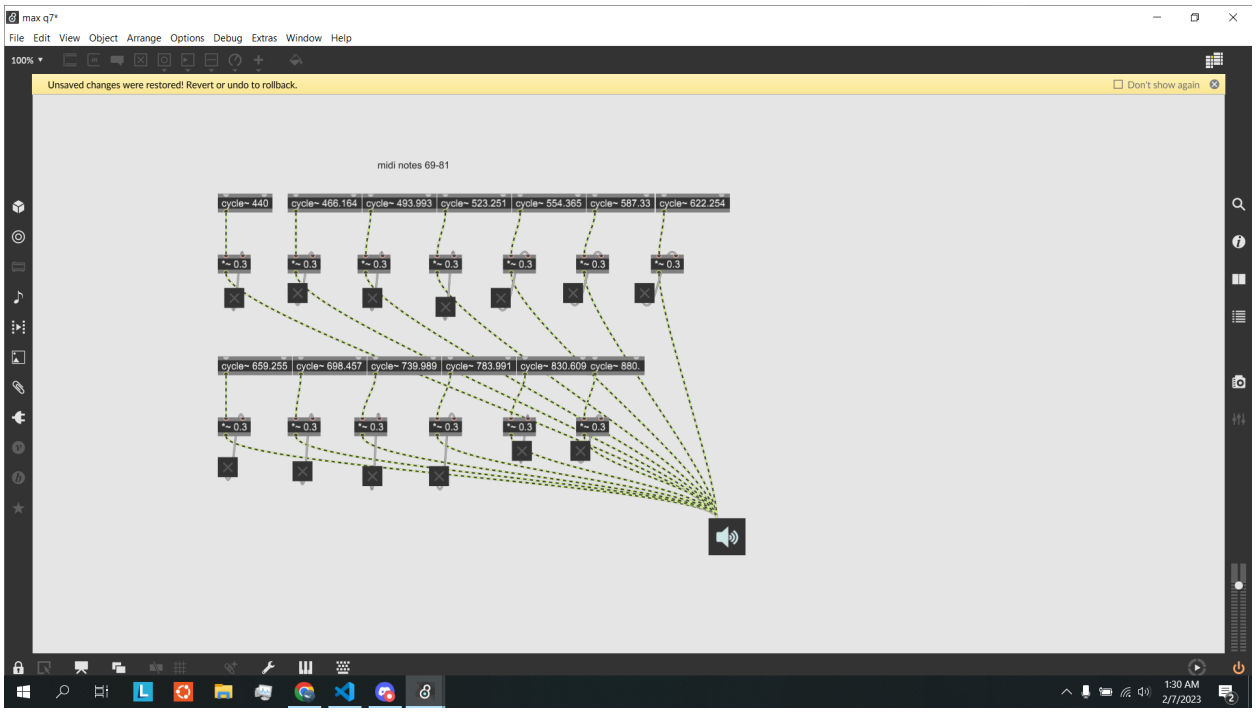


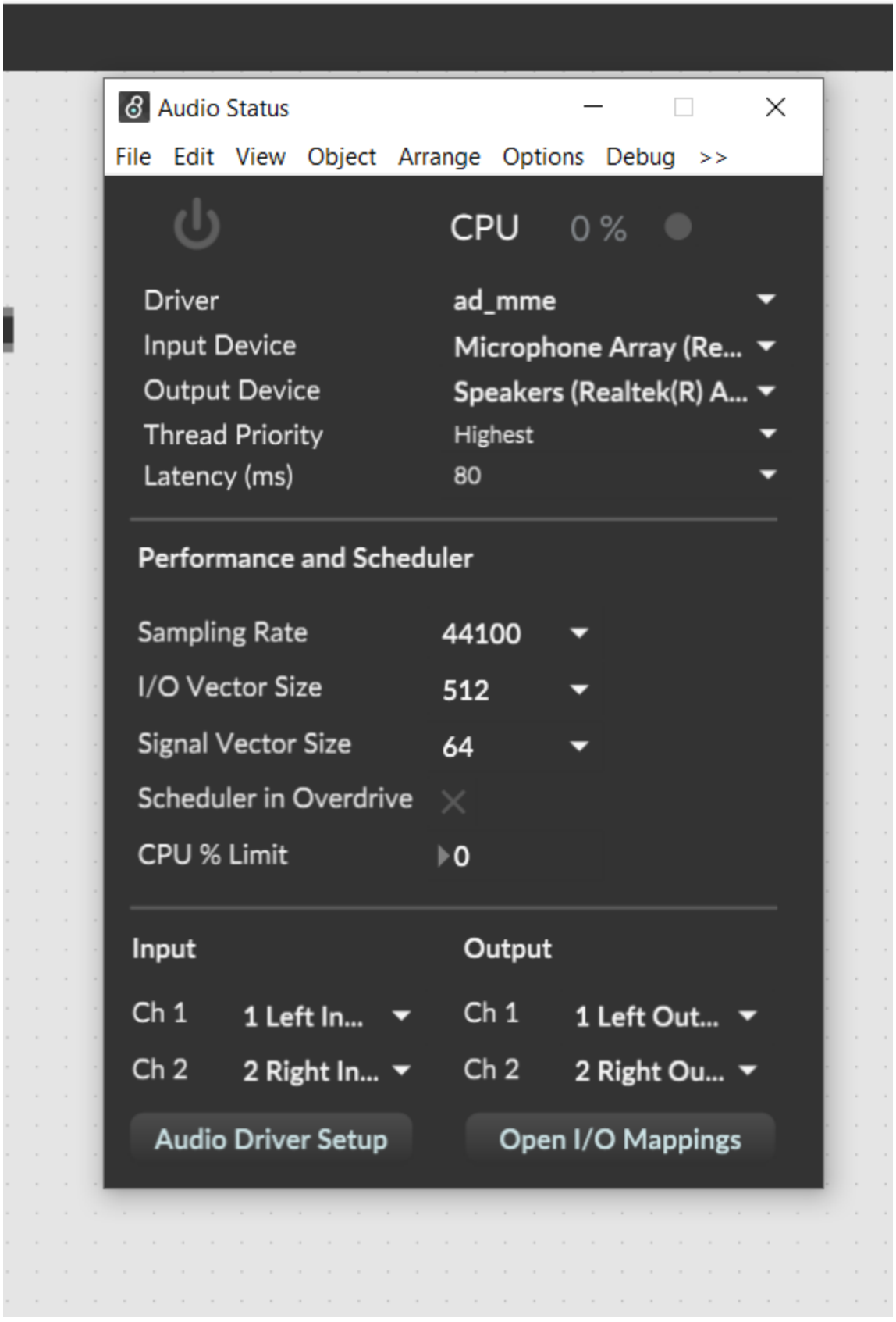


For question 6 i had to make my program support polyphony and multiple oscillators. I implemented polyphony using a fixed number of oscillators (one for each voice - 2). Then I checked to see what level of polyphony I can support with your implementation on my laptop until it could not keep up with real time and the sound did not sound as one. I created functions for outputting different files using individual notes/durations as input in order to play them at the same time and get different notes combined together. Then I have a different function that will play the sounds together for their possibly

different durations. To see how many oscillators i could play at once without distorting the thought of real time I used a file that would beep once every other second so that I could register the distortion in sound. I was able to run 3 files at the same time without noticing but once I ran the 4th i noticed a small sound difference, it sounded like the beeps were vibrating a little bit which is a bad description but im not sure how to describe it.

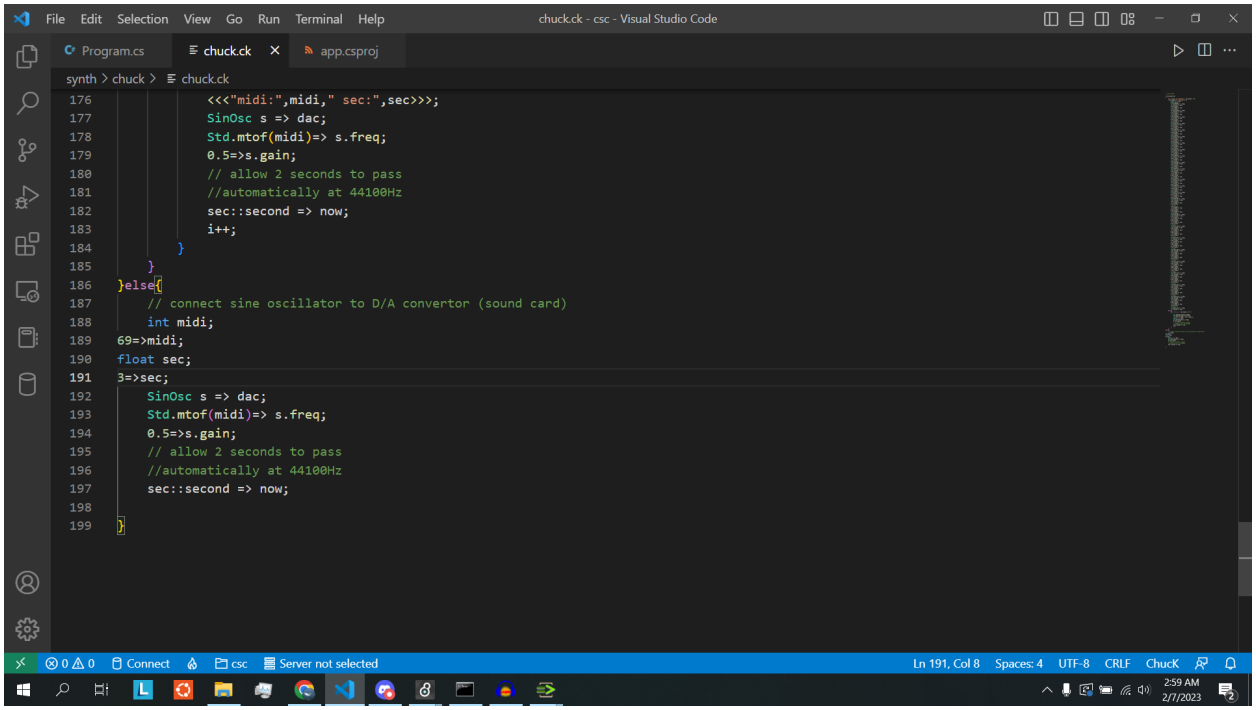


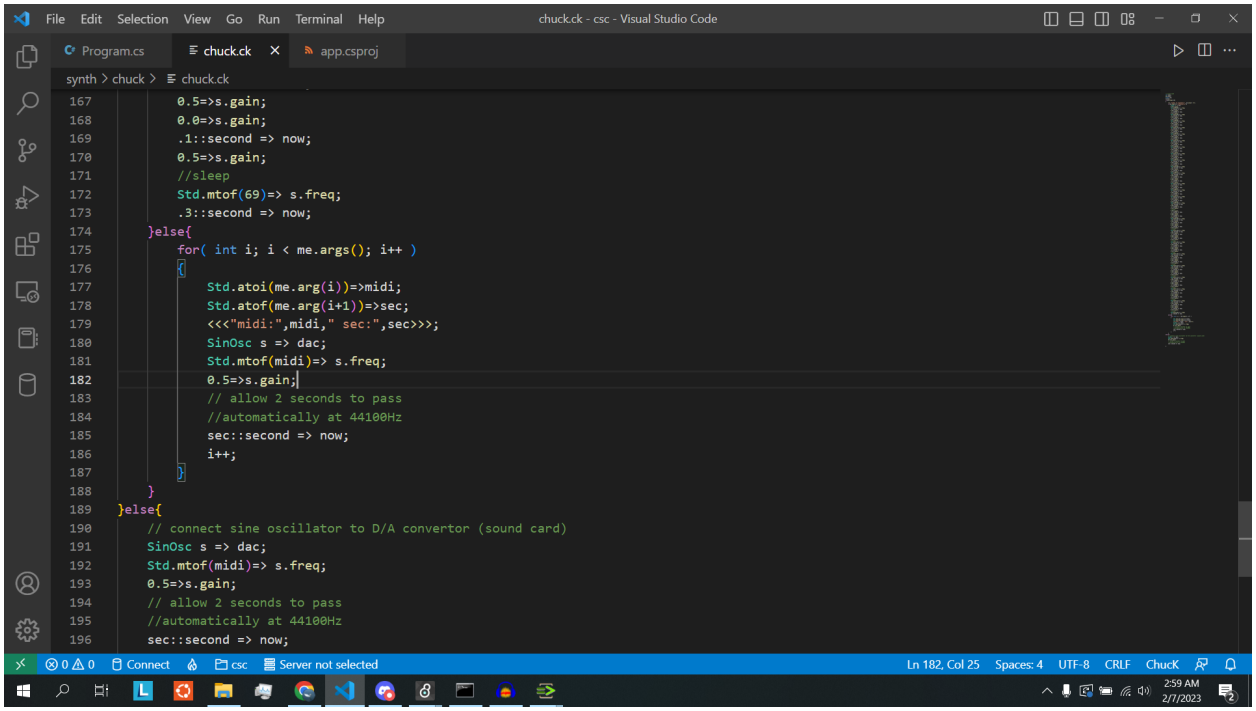


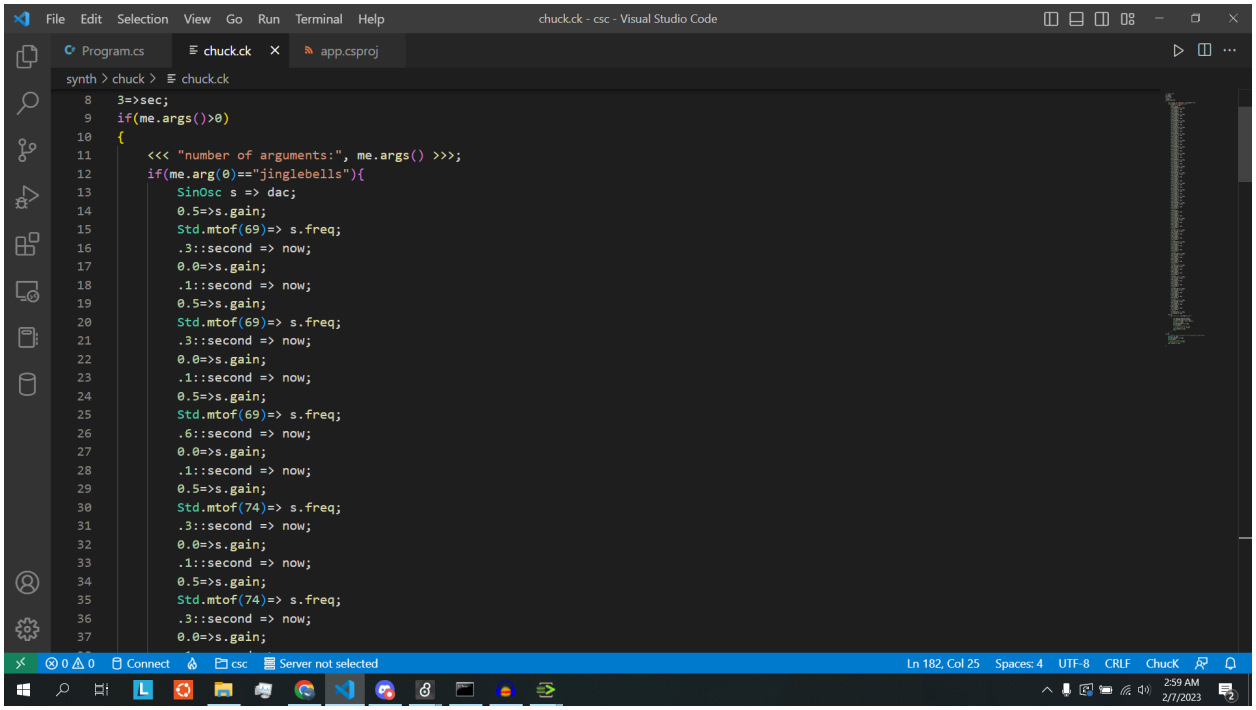

Audio Status
File Edit View Object Arrange Options Debug >>
CPU 0 %
Driver ad_mme
Input Device Microphone Array (Re...
Output Device Speakers (Realtek(R) A...
Thread Priority Highest
Latency (ms) 80
Performance and Scheduler
Sampling Rate 44100
I/O Vector Size 512
Signal Vector Size 64
Scheduler in Overdrive
CPU % Limit 0
Input
Output
Ch 1 1 Left In...
Ch 2 2 Right In...
Ch 1 1 Left Out...
Ch 2 2 Right Ou...
Audio Driver Setup
Open I/O Mappings

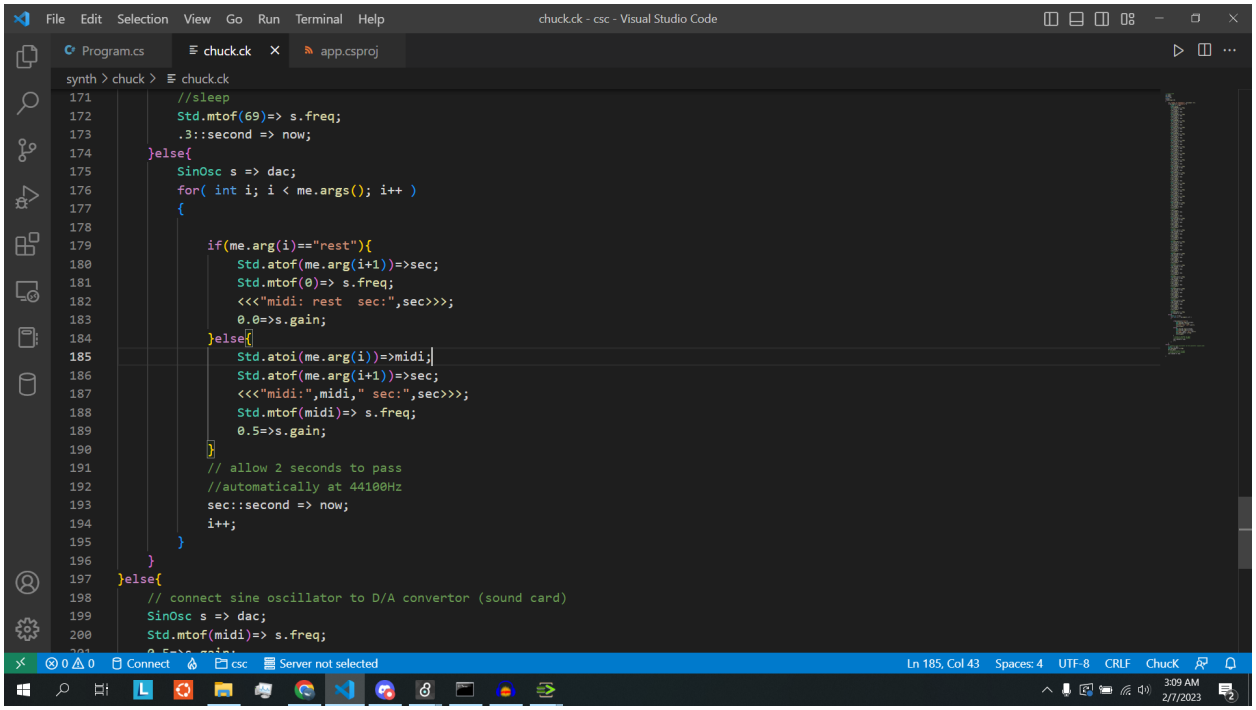
Pitch	Frequency [Hz]	MIDI
A4	440.000	69
A#4	466.164	70
B4	493.883	71
C5	523.251	72
C#5	554.365	73
D5	587.330	74
D#5	**622.254**	75
E5	659.255	76
F5	698.457	77
F#5	739.989	78
G5	783.991	79
G#5	830.609	80
A5	880.000	81

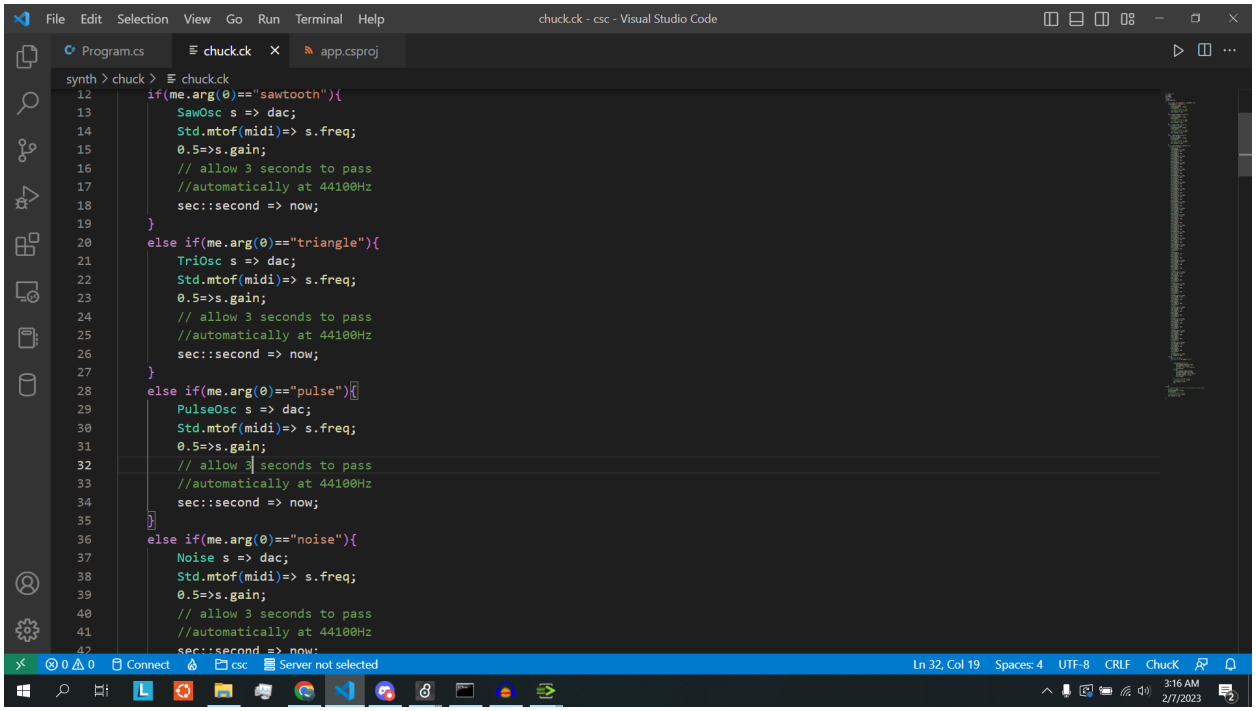
For question 7 I had to do questions 1 and 2 using max/msp. The first question is shown in the first screenshot where i used cycle~ at a frequency of 440 in order to play a sound. This sound was loud so I lowered the volume using *~ .3. The sample rate is by default 44100 as shown in screenshot 3. For question 2 i used 13 different notes shown in screenshot 4 with their corresponding frequencies they call connect to 1 audio output so I used a toggle button that gives either 0 or 1 to the volume essentially muting the sound until i press the buttons to play notes individually or together. I am able to play a simple melody(jingle bells) by switching through different toggle buttons turning the sound on and off.


File Edit Selection View Go Run Terminal Help
chuck.ck - csc - Visual Studio Code
Program.cs
chuck.ck
app.csproj
synth > chuck > chuck.ck
<<<"midi:",midi," sec:",sec>>>;
SinOsc s => dac;
Std.mtof(midi)=> s.freq;
0.5=>s.gain;
// allow 2 seconds to pass
//automatically at 44100Hz
sec::second => now;
i++;
}
}
}else{
// connect sine oscillator to D/A convertor (sound card)
int midi;
69=>midi;
float sec;
3=>sec;
SinOsc s => dac;
Std.mtof(midi)=> s.freq;
0.5=>s.gain;
// allow 2 seconds to pass
//automatically at 44100Hz
sec::second => now;
}
Connect
csc
Server not selected
Ln 191, Col 8
Spaces: 4
UTF-8
CRLF
ChucK
2:59 AM
2/7/2023


File Edit Selection View Go Run Terminal Help
chuck.ck - csc - Visual Studio Code
Program.cs
chuck.ck
app.csproj
synth > chuck > chuck.ck
0.5=>s.gain;
0.0=>s.gain;
.1::second => now;
0.5=>s.gain;
//sleep
Std.mtof(69)=> s.freq;
.3::second => now;
}else{
for(int i; i < me.args(); i++)
{
Std.atoi(me.arg(i))=>midi;
Std.atof(me.arg(i+1))=>sec;
<<<"midi:",midi," sec:",sec>>>;
SinOsc s => dac;
Std.mtof(midi)=> s.freq;
0.5=>s.gain;
// allow 2 seconds to pass
//automatically at 44100Hz
sec::second => now;
i++;
}
}
}else{
// connect sine oscillator to D/A convertor (sound card)
SinOsc s => dac;
Std.mtof(midi)=> s.freq;
0.5=>s.gain;
// allow 2 seconds to pass
//automatically at 44100Hz
sec::second => now;
Connect
csc
Server not selected
Ln 182, Col 25
Spaces: 4
UTF-8
CRLF
ChucK
2:59 AM
2/7/2023

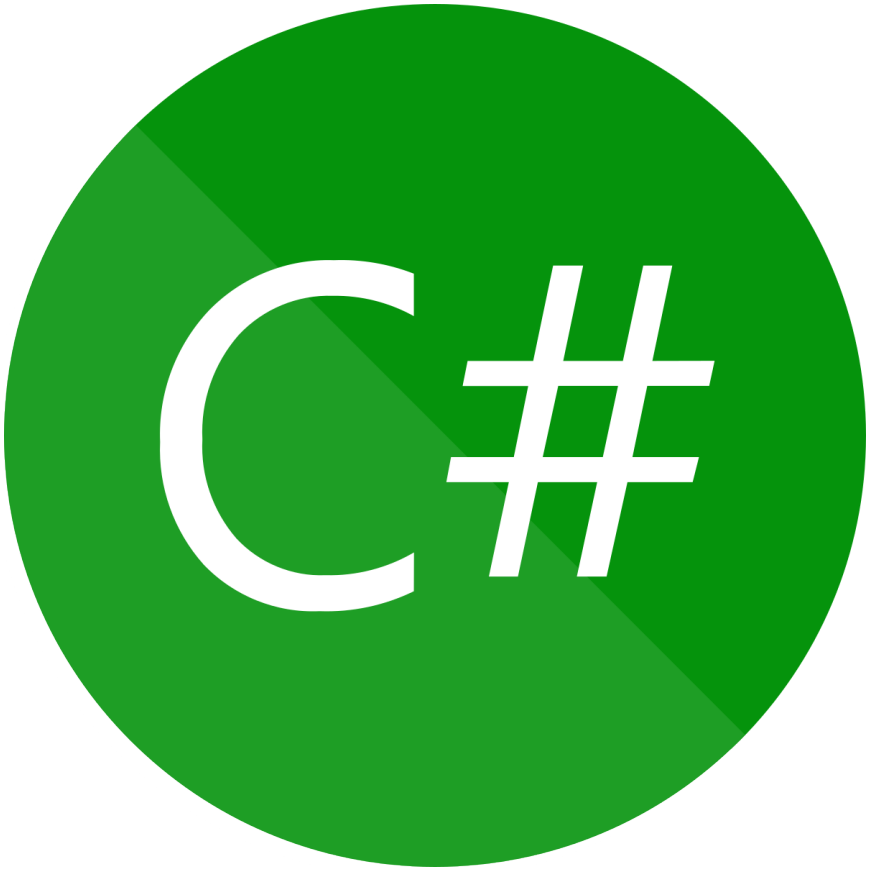

File Edit Selection View Go Run Terminal Help
chuck.ck - csc - Visual Studio Code
Program.cs
chuck.ck
app.csproj
synth > chuck > chuck.ck
3=>sec;
if(me.args()>0)
{
<<< "number of arguments:", me.args() >>>;
if(me.arg(0)=="jinglebells"){
SinOsc s => dac;
0.5=>s.gain;
Std.mtof(69)=> s.freq;
.3::second => now;
0.0=>s.gain;
.1::second => now;
0.5=>s.gain;
Std.mtof(69)=> s.freq;
.3::second => now;
0.0=>s.gain;
.1::second => now;
0.5=>s.gain;
Std.mtof(69)=> s.freq;
.6::second => now;
0.0=>s.gain;
.1::second => now;
0.5=>s.gain;
Std.mtof(74)=> s.freq;
.3::second => now;
0.0=>s.gain;
.1::second => now;
0.5=>s.gain;
Std.mtof(74)=> s.freq;
.3::second => now;
0.0=>s.gain;
0 0 Connect csc Server not selected
Ln 182, Col 25 Spaces: 4 UTF-8 CRLF ChucK
2:59 AM
2/7/2023



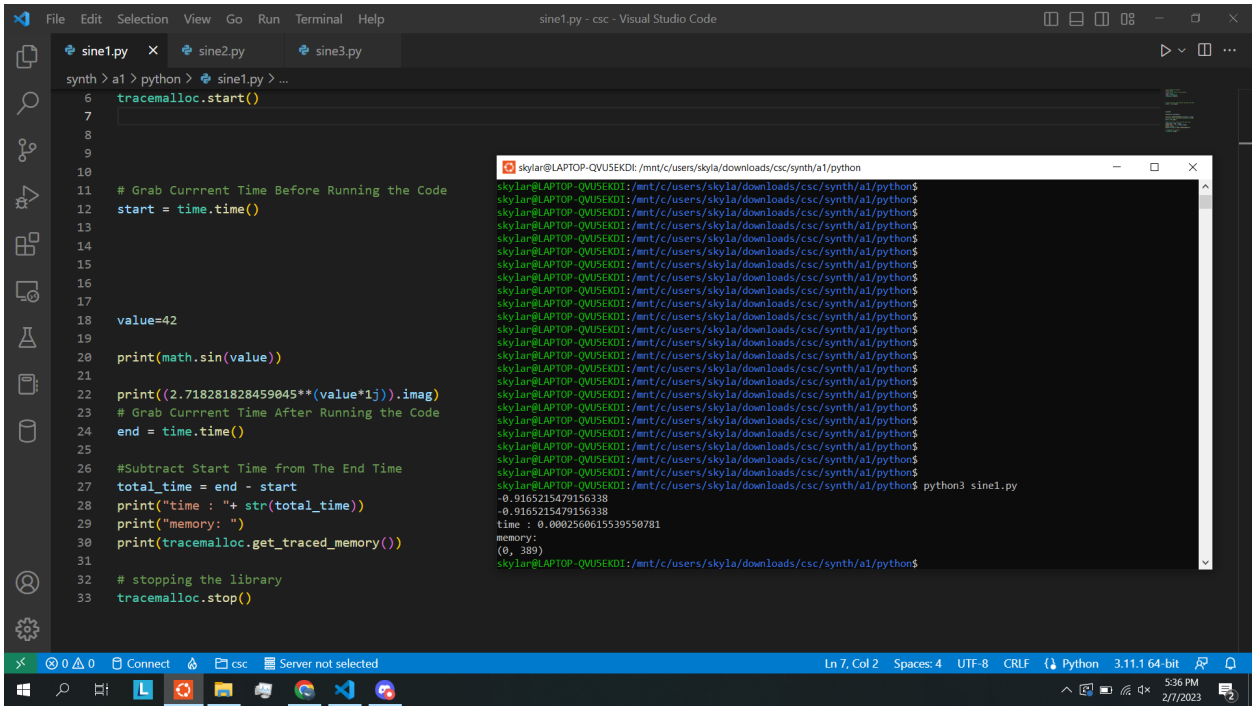


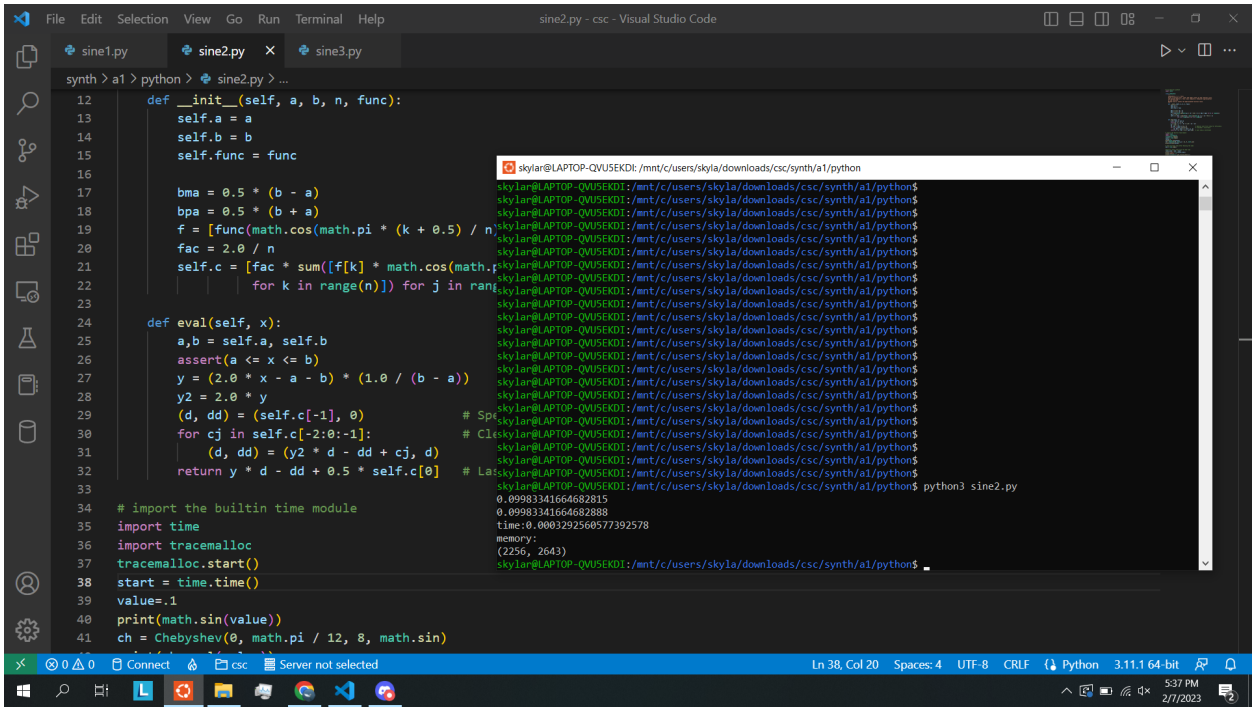
For question 8 i had to re-Implement questions 1,2,3,4 using ChucK. Question 1 is shown in screenshot 1 using SinOsc we can play sound by definition freq and gain then running the x::second => now to define how many seconds we want to use. Question 2 is shown in both screenshot 2 and screenshot 3 where i play the jingle bells simple melody and allow the code to take command line argument to play a midi note for a duration. Question 3 is shown in screenshot 4 where it will take command line arguments to play rest for x duration which is just oscillator playing sound for x seconds

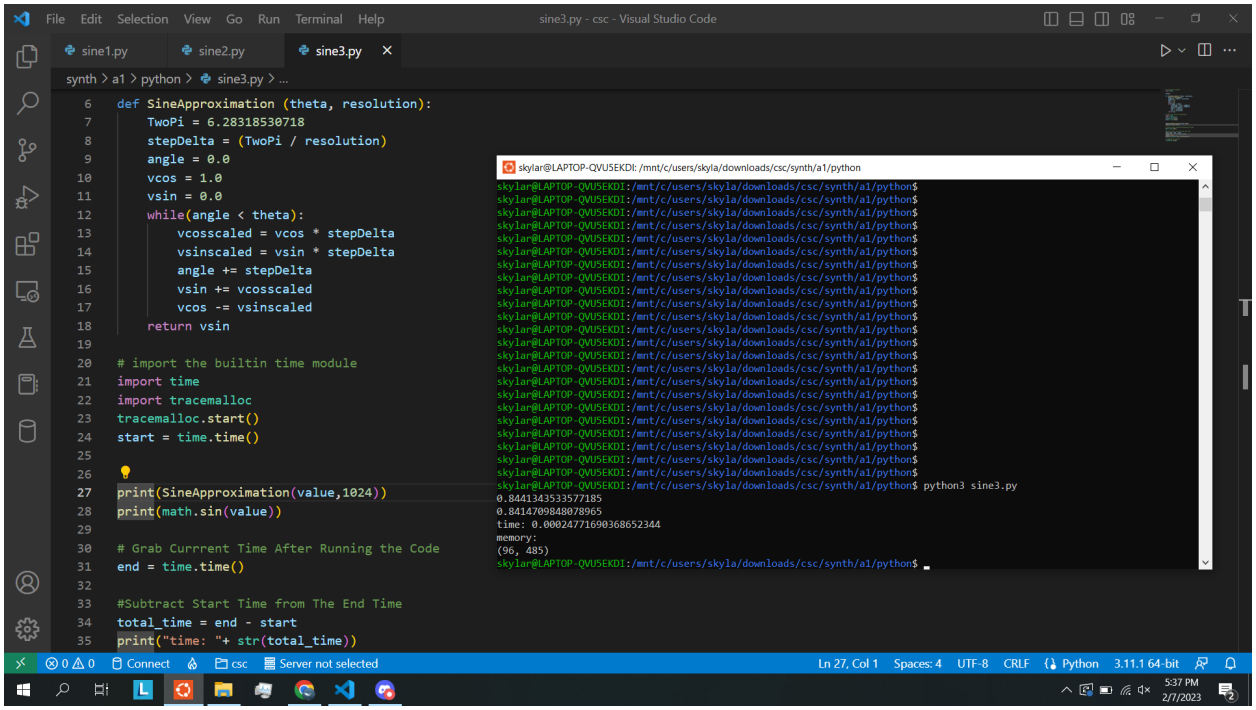
on mute and it will also take many notes/rests in succession and will play them in order. Question 4 is shown in screenshot 5 and will play different Oscillators (sawtooth, noise, pulse, triangle) using different Osc variables.



Question 9 - For this assignment I started using haskell which is a language i had never used before and it was very confusing because I was getting errors when other people also using haskell were not getting the same errors i was with the same code. This was frustrating for me so after finishing question 1 in haskell i swapped to c# to do question 9 because even though I had never coded/used before just like haskell, I had used C so i thought it might be similar.


File Edit Selection View Go Run Terminal Help
sine1.py - csc - Visual Studio Code
sine1.py
sine2.py
sine3.py
synth > a1 > python > sine1.py > ...
tracemalloc.start()
# Grab Currrent Time Before Running the Code
start = time.time()
value=42
print(math.sin(value))
print((2.718281828459045**(value*1j)).imag)
# Grab Currrent Time After Running the Code
end = time.time()
#Subtract Start Time from The End Time
total_time = end - start
print("time : "+ str(total_time))
print("memory: ")
print(tracemalloc.get_traced_memory())
# stopping the library
tracemalloc.stop()
skylar@LAPTOP-QVU5EKDI: /mnt/c/users/skyla/downloads/csc/synth/a1/python
skylar@LAPTOP-QVU5EKDI:/mnt/c/users/skyla/downloads/csc/synth/a1/python$ python3 sine1.py
-0.9165215479156338
-0.9165215479156338
time : 0.0002560615539550781
memory:
(0, 389)
skylar@LAPTOP-QVU5EKDI:/mnt/c/users/skyla/downloads/csc/synth/a1/python$
Ln 7, Col 2 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
5:36 PM
2/7/2023


File Edit Selection View Go Run Terminal Help
sine2.py - csc - Visual Studio Code
sine1.py
sine2.py
sine3.py
synth > a1 > python > sine2.py > ...
def __init__(self, a, b, n, func):
self.a = a
self.b = b
self.func = func
bma = 0.5 * (b - a)
bpa = 0.5 * (b + a)
f = [func(math.cos(math.pi * (k + 0.5) / n
fac = 2.0 / n
self.c = [fac * sum([f[k] * math.cos(math.
for k in range(n)]) for j in ran
def eval(self, x):
a,b = self.a, self.b
assert(a <= x <= b)
y = (2.0 * x - a - b) * (1.0 / (b - a))
y2 = 2.0 * y
(d, dd) = (self.c[-1], 0)
for cj in self.c[-2:0:-1]:
(d, dd) = (y2 * d - dd + cj, d)
return y * d - dd + 0.5 * self.c[0]
# import the builtin time module
import time
import tracemalloc
tracemalloc.start()
start = time.time()
value=.1
print(math.sin(value))
ch = Chebyshev(0, math.pi / 12, 8, math.sin)
skylar@LAPTOP-QVU5EKDI: /mnt/c/users/skyla/downloads/csc/synth/a1/python
skylar@LAPTOP-QVU5EKDI:/mnt/c/users/skyla/downloads/csc/synth/a1/python$ python3 sine2.py
0.09983341664682815
0.09983341664682888
time:0.0003292560577392578
memory:
(2256, 2643)
skylar@LAPTOP-QVU5EKDI:/mnt/c/users/skyla/downloads/csc/synth/a1/python$
Ln 38, Col 20 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
5:37 PM
2/7/2023



For question 10 I got 3 different ways of calculating sine in python without the math.sin() function. I calculated the time each file took using the time import and the memory using the tracemalloc import. For the first sine file I used the formula of 2.718281828459045**(x*1j)).imag to calculate sine of a random number (in this case 42). In the print statement the first value is the true value of sine(42), the second value is the value of the formula(42), the third value is the amount of time it took, and the fourth/fifth line is the amount of memory used. For the second sine file I used the formula of Chebyshev to calculate sine of a random number (in this case 0.1). In the print statement the first value is the true value of sine(0.1), the second value is the value of the ch.eval(0.1), the third value is the amount of time it took, and the fourth/fifth line is the amount of memory used. For the third sine file I used the formula of sine approximation using 2Pi, step, angle, precos, theta, and presin in a combination using a while loop in order to calculate sine of a random number (in this case 1). In the print statement the first value is the true value of SineApproximation(1), the second value is the value of the sine(1), the third value is the amount of time it took, and the fourth/fifth line is the amount of memory used. In terms of time taken for execution the order for least to greatest time taken is third sine file, first sine file, second sine file. In terms of memory used for execution the order for least to greatest memory used is first sine file, third sine file, second sine file.