University of Pennsylvania
Department of Electrical and System Engineering
Digital Audio Basics

ESE150, Spring 2022 Final Solutions Wednesday, May 4

- Exam ends at 11:00AM; begin as instructed (target 9:00AM)
- Do not open exam until instructed to begin exam.
- Problems weighted as shown.
- Calculators allowed.
- Closed book = No text or notes allowed.
- Provided reference materials on next to last page.
- Show work for partial credit consideration.
- Unless otherwise noted, answers to two significant figures are sufficient.
- Sign Code of Academic Integrity statement (see last page for code).

I certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this exam.

Name: Solutions

1			2			3	4					5				6	7		8
a	b	c	bw	d	r		a	b	c	d	e	a	b	c	d		a	b	
4	4	4	10	5	5	12	2	2	3	3	5	5	2	2	2	10	5	5	10

1. Consider a remote monitoring microphone taking 16b audio samples at 10KHz.

   (a) What is the highest frequency this sampling rate can represent?
   5KHz – Nyquist frequency at half the sample rate.

   (b) Processing the samples in time windows of 1024 samples, how many multiplies are required to extract a single frequency component?
   2048 – we must perform a dot product against the sin and cosine time components for the target frequency.

   (c) Assuming the target is to extract 512 frequencies from each window, how many multiplies per second must be processed to convert the windows into the frequency domain and keep up with the real-time input rate?
   $$\frac{2048 \times 512 \,\text{multiplies}}{1024 \,\text{samples} \times 10^{-4}\,\text{s/sample}} = 10,240,000$$
   $\approx$ 10M multipies/second

2. Continuing with the remote monitor microphone, consider sending "interesting" time window samples (same 1024 sample windows at 10KHz).

   - Threshold out amplitudes below a set minimum amplitude per frequency (configurable on a per microphone basis).
   - After filtering, we expect most time windows (>90%) to not include any frequencies and the remaining time windows to contain only a few interesting frequencies (20 on average).
   - time-of-window is 32b; frequency needs enough bits to identify 512 frequencies; assume time and frequency amplitudes need 16b.
   - For the following, assume each packet needs 96b for source/sink headers and checksum.

   For each of the following estimate the expected bandwidth requirement and use to rank from least bandwidth (1) to most (5). Also describe the impact of dropped and reordered packets (Assuming the network may drop or reorder packets, what errors may this introduce in waveform reconstruction and playback? Assume reconstruction program tries to do the best it can with the data it does receives.)

   (a) send each non-thresholded frequency in a packet as a single triple (time-of-window, frequency, amplitude)

Bandwidth Calculation	$(96 + 32 + 9 + 16) \times 20 \times 0.1 = 306$
Rank	3
Drop Impact	only lose one frequency
Reorder Impact	none; time-of-window allows proper ordering

(b) if there are any non-thresholded frequencies in a time window, compute a new time sequence removing the thresholded frequencies. Send a packet with a single time-of-window identifier for the packet and all the time samples for the new time sequence for the window.

Bandwidth Calculation	$(96 + 32 + 1024 \times 16) \times 0.1 = 1651.2$
Rank	4
Drop Impact	lose entire frame
	$= 0.1$ seconds of audio
Reorder Impact	none;
	time-of-window allows proper ordering

(c) if there are any non-thresholded frequencies in a time window, send all the non-thresholded frequencies in a packet with a single time-of-window identifier and a set of non-thresholded (frequency, amplitude) pairs

Bandwidth Calculation	$(96 + 32 + 20 \times (9 + 16)) \times 0.1 = 62.8$
Rank	1
Drop Impact	lose entire frame
	$= 0.1$ seconds of audio
Reorder Impact	none;
	time-of-window allows proper ordering

(d) for every time window, send time samples for window in a packet after removing the thresholded frequencies

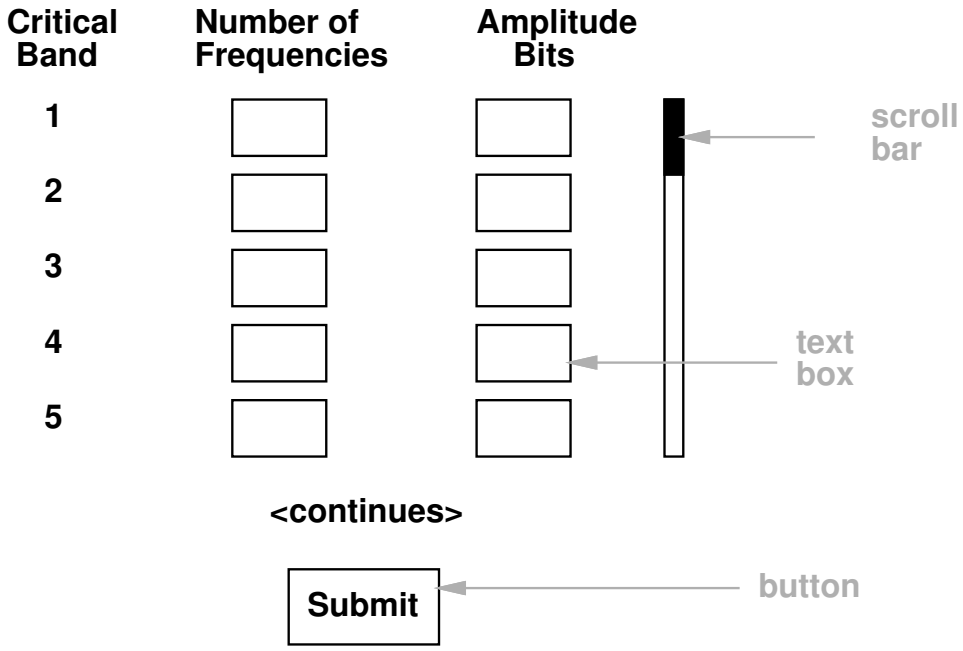
Bandwidth	$96 + 1024 \times 16 = 16480$
Calculation	
Rank	5
Drop Impact	lose entire frame;
	now other frames are interpretted in wrong time window
Reorder Impact	reconstruction remains out-of-order

(e) for every time window, send all the non-thresholded frequencies in a packet as a (possibly empty) set of non-thresholded (frequency, amplitude) pairs

Bandwidth	$96 + (9 + 16) \times 20 \times 0.1 = 146$
Calculation	
Rank	2
Drop Impact	lose entire frame;
	now other frames are misinterpretted in wrong time window
Reorder Impact	reconstruction remains out-of-order

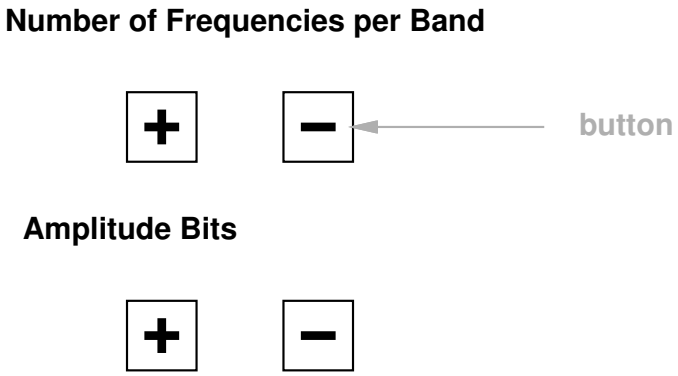
3. Working with roughly the same remote monitoring scenario as the first two problems, consider the following user interfaces to control the quality-bandwidth tradeoffs. For each of the following user interfaces for this task: (a) rank their ease-of-use from (1) easiest to (4) hardest to use; (b) identify strengths and weaknesses (at least one of each) [hint: general public is unlikely to be familiar with concepts like critical bands and frequency domain.]:

   (a) User sets the number of frequencies to keep per band and quantization-level per band.



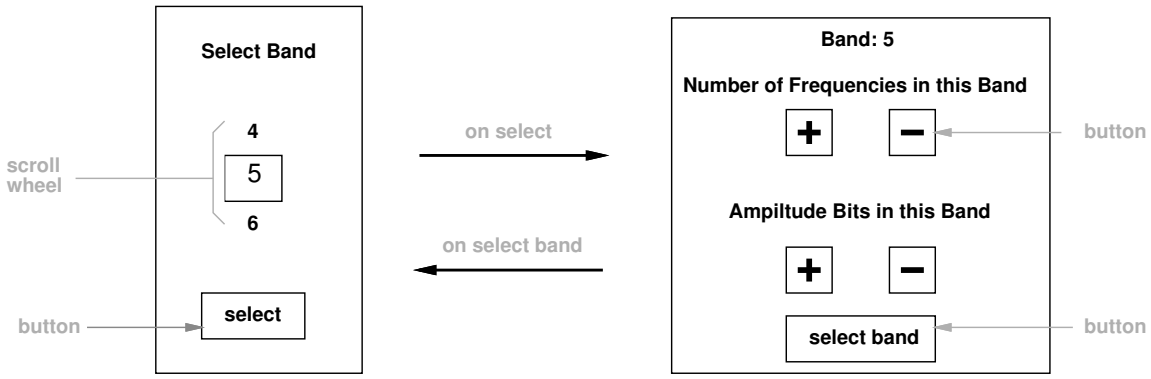
Ease-of-Use Rank	4
Strength	precise control can be selected directly
Weakness	complicated interface requiring the input of a large amount of information (significant typing)
	most users not understand signfigance of bands and numbers
	inputs not constrained to legal and meaningful values

(b) User has one control to increase or decrease frequencies per band and one to increase/decrease quantization-level per band; the single setting for frequencies and quantization apply to all bands.



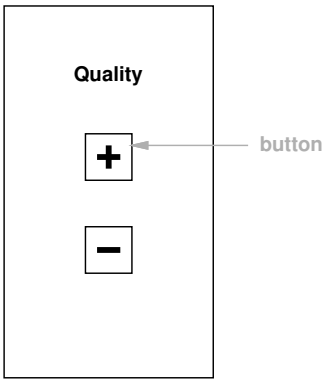
Ease-of-Use Rank	2
Strength	simple interface that does control quality
Weakness	cannot achieve best quality-bandwidth tradeoff since some bands are less important than others, but all are treated the same here
	Not give any indication of current quality level and how close to limits.

(c) User has one control to select a band, then two controls to increase/decrease frequencies in the band and another to increase or decrease quantization-level per in the band.



Ease-of-Use Rank	3
Strength	controls per band allowing different frequencies and bits in band as we know is important to get best quality for a given bandwidth
Weakness	Need to control quality for 24 bands will be tedious and require many key presses
	most user do not understand significance of the various bands; the extra control will mostly be confusing to the user.

(d) User has a single control to increase or decrease the quality; each time the increase (decrease) button is pressed the quality makes a small step in the indicated direction.



Ease-of-Use Rank	1
Strength	simple, intuitive interface in terms of what user understands (quality)
Weakness	may need to press +/- many times to achieve desired quality
	Not show any indication of where quality is between limits.
	(updating quality would also serve to show response to keypress)

4. Working with the input from the UI described in case (d) in the previous problem, what does the program need to do for each user input to increase or decrease the quality? Assume the current state of the compression is represented internally like case (a) with a number of frequencies to keep per band and a quantization level for each band:

```
int num_freqs_band[24];
int bits_band[24];
```

   (a) Describe the ways in which the current internal compression control state can be changed in response to an increase quality request?
   Increase `num_freqs_band` for some band; or increase `bits_band` for some band.

   (b) For your answer to (a), for a given increase request, count the number of changes the program has to choose from? $48 = 24 \times 2$

   (c) How do you compute the number of bits required for a time sample window given a particular internal compression control state? (give an equation)
   let freqs_in_band[b] be the number of total frequencies available to select from in band b.

$$bits = \sum_{b=1}^{24} (num\_freqs\_band[b] \times (\lceil \log_2(freqs\_in\_band[b]) \rceil + bits\_band[b])) \quad (1)$$

   (d) How could you compute the error introduced for a particular internal compression control state?
   Best for psychoacoustics would be to compute error in frequency domain based on human significance of frequencies. Assign each frequency, f, a significance, $Significance[f]$. Let FA_{ref} be the intended amplitude for each frequency. Let FA_{encode} be the effective encoded amplitude based on frequency presence and bits assigned.

$$Error = \sum_{f=0}^{511} (|FA_{encode}[f] - FA_{ref}[f]| \times Significance[f]) \quad (2)$$

Will also accept time domain errors. Let t_{encode} = time sequence reconstructed from current encoding. Let t_{ref} = intended time sequence.

$$Error = \sum_{s=0}^{1023} |t_{encode} - t_{ref}| \tag{3}$$

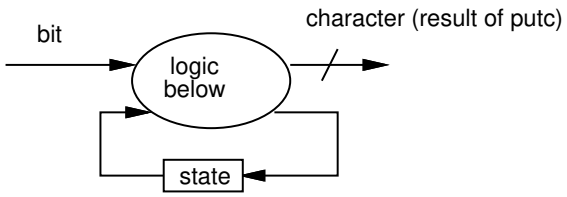
(e) Given a user request to increase the quality, describe an algorithm to determine which of the changes from (a), (b) to make to the current internal compression control state?

- For each of the 48 potential increase operations, o
  - compute the increase in bits required by operation o using equation from (c)
  - compute the decrease in error by operation o using equation from (d)
  - compute the benefit ratio for o by dividing the error decrease by the bandwidth increase
- pick the o^* that has the largest benefit ratio
- apply the o^* change to the internal compression control state

The simplest case is to perform this analysis on the next window to encode after the increase request. However, we expect the frequency content to vary over time, so a single time window may not be representative. In fact, the user likely signalled the need for a quality increase based on an earlier set of time windows. So, it may be better to use multiple previous windows in the assessment.

Will allow other formulations to pick quality increase, including just using decrease in errors.

5. Consider the following FSM for decoding serial, Huffman encoded data into decimal digits.




```
int state=START;
while(true) {
  int bit=nextInputBit();
  switch(state) {
  case START:  if (bit==0) state=S0; else state=S1; break;
  case S0:  if (bit==0) {state=START; putc('0');} else {state=START; putc('1');}
            break;
  case S1:  if (bit==0) state=S10; else state=S11; break;
  case S10:  if (bit==0) {state=START; putc('2');} else {state=START;putc('3');}
             break;
  case S11:  if (bit==0) state=S110; else state=S111; break;
  case S110:  if (bit==0) {state=START; putc('4');} else {state=START; putc('5');}
              break;
  case S111:  if (bit==0) state=S1110; else state=S1111; break;
  case S1110:  if (bit==0) {state=START; putc('6');} else {state=START; putc('7');}
               break;
  case S1111:  if (bit==0) {state=START; putc('8');} else {state=START; putc('9');}
               break;
               }
  }
```

(a) What is the encoding for each of the digits?

letter	encoding
0	00
1	01
2	100
3	101
4	1100
5	1101
6	11100
7	11101
8	11110
9	11111

(b) Using the encoding, how many bits does it take to encode the 4 digit string: 1500
2+4+2+2=10

(c) Using the encoding, how many bits does it take to encode the 4 digit string: 9876
5+5+5+5=20

(d) If you used a unified encoding for the digits, how many bits would it require to encode a 4 digit string?
$\lceil \log_2(10) \rceil \times 4 = 16$

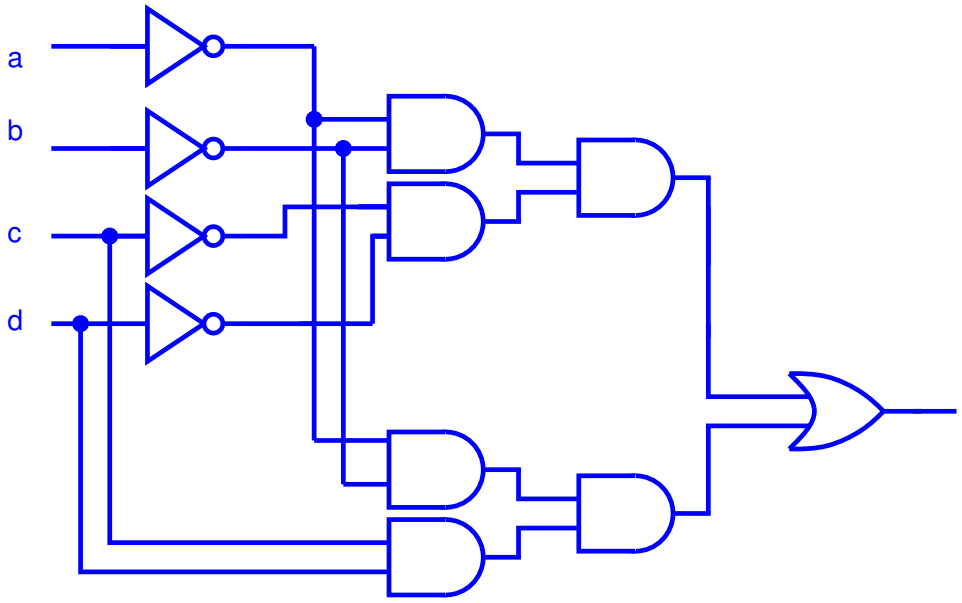
6. Implement the following truth table using inverters and 2-input AND and OR gates.

a	b	c	d	out
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

out=/a*/b*/c*/d+/a*/b*c*d

```
out=OR2(
        AND2(
          AND2(NOT(a),NOT(b)),
          AND2(NOT(c),NOT(d))),
        AND2(
          AND2(NOT(a),NOT(b)),
          AND2(c,d)))
```

(This page left mostly blank for pagination.)



7. If a processor cannot keep up with the current set of computations (processes) that it is running, we might consider moving its computation to an offload processor in the cloud.

   (a) What information needs to be sent in order for the computation to resume on an offload processor that is accessible over the network and get the same result as if the computation had remained on the original processor?

   i. program counter

   ii. contents of instruction memory

   iii. contents of data memory

   (b) How can this information be compressed to reduce network bandwidth when the computation is sent to the offload processor? (What kinds of compression, if any, are appropriate? How effective are they likely to be? Assume the memory for the process is initialized to zero before the computation starts.)

   It must use a lossless compression scheme.

   Huffman encoding would be losseless and could be useful.

   - Instruction memory is not likely to be random, so may compress based on more frequent instructions.
   - It's also possible there is regularity in the data memory data that also allows some compression.
   - If there is much unused instruction or data memory, since it is initialized to zero, it will get a short encoding and be compressible. (It is also possible to simply not send regions of memory that are all zeros.)
     - Not all programs will fill all memory.
     - At the beginning of execution, most of the data memory will be at its initialized value of zero.

(This page left mostly blank for pagination.)

8. You are developing a new compression technique for code memory that could allow more compact executables and more compact storage and transmissions of the code portion of a process. If you were to prepare a US patent application, is there some chance it would be patentable:

   (a) when you first have the abstract idea?
   No – abstract ideas are not patentable.

   (b) after you have an implementation?
   Yes – an implementation means you have reduced the invention to practice.

   (c) before publishing the technique on a blog?
   Yes – US filing must be within one year of first public disclosure. Best to file before any public disclosure (and essential for non-US patents).

   (d) after you read a patent application on the same technique?
   No – Patents now go to first to file.

   (e) if your technique is a simple application of Huffman coding?
   No – The invention must be non-obvious to one skilled in the art. A simple application of Huffman coding would be obvious to anyone familiar with compression.

Human auditory critical bands:

Band Number	Low	High
1	20	100
2	100	200
3	200	300
4	300	400
5	400	510
6	510	630
7	630	720
8	720	920
9	920	1080
10	1080	1370
11	1270	1480
12	1480	1720
13	1720	2000
14	2000	2320
15	2320	2700
16	2700	3150
17	3150	3700
18	3700	4400
19	4400	5300
20	5300	6400
21	6400	7700
22	7700	9500
23	9500	12000
24	12000	15500

Code of Academic Integrity

Since the University is an academic community, its fundamental purpose is the pursuit of knowledge. Essential to the success of this educational mission is a commitment to the principles of academic integrity. Every member of the University community is responsible for upholding the highest standards of honesty at all times. Students, as members of the community, are also responsible for adhering to the principles and spirit of the following Code of Academic Integrity.*

Academic Dishonesty Definitions

Activities that have the effect or intention of interfering with education, pursuit of knowledge, or fair evaluation of a student's performance are prohibited. Examples of such activities include but are not limited to the following definitions:

A. Cheating Using or attempting to use unauthorized assistance, material, or study aids in examinations or other academic work or preventing, or attempting to prevent, another from using authorized assistance, material, or study aids. Example: using a cheat sheet in a quiz or exam, altering a graded exam and resubmitting it for a better grade, etc.

B. Plagiarism Using the ideas, data, or language of another without specific or proper acknowledgment. Example: copying another person's paper, article, or computer work and submitting it for an assignment, cloning someone else's ideas without attribution, failing to use quotation marks where appropriate, etc.

C. Fabrication Submitting contrived or altered information in any academic exercise. Example: making up data for an experiment, fudging data, citing nonexistent articles, contriving sources, etc.

D. Multiple Submissions Multiple submissions: submitting, without prior permission, any work submitted to fulfill another academic requirement.

E. Misrepresentation of academic records Misrepresentation of academic records: misrepresenting or tampering with or attempting to tamper with any portion of a student's transcripts or academic record, either before or after coming to the University of Pennsylvania. Example: forging a change of grade slip, tampering with computer records, falsifying academic information on one's resume, etc.

F. Facilitating Academic Dishonesty Knowingly helping or attempting to help another violate any provision of the Code. Example: working together on a take-home exam, etc.

G. Unfair Advantage Attempting to gain unauthorized advantage over fellow students in an academic exercise. Example: gaining or providing unauthorized access to examination materials, obstructing or interfering with another student's efforts in an academic exercise, lying about a need for an extension for an exam or paper, continuing to write even when time is up during an exam, destroying or keeping library materials for one's own use., etc.

* If a student is unsure whether his action(s) constitute a violation of the Code of Academic Integrity, then it is that student's responsibility to consult with the instructor to clarify any ambiguities.