Lab 10 grading guidelines

Pre-lab

Out of 10 points. This lab part **is** expected to compile.

- 8 points: Huffman encoding
    - 2 points for each correct test run (there are 4 test runs). If their code is *more* efficient than ours, then they still get this point, as long as it's a valid translation.
- 2 points: Section 3 of the encoded file
    - This section should list: the compression ratio and the cost of the Huffman tree. 1 point for each that is computed correctly (again, note that their compression may be more efficient than our solutions)
- If they did not build the Huffman tree, they get a 0 for the pre-lab
- If they did build a Huffman tree but their implementation did not work, they get 3 points

Pre-lab input

The four input files provided were passed as input to the submitted in-lab code.

The input for each of the files, shown in the normal?.txt files, is listed below. Note that we ignore newlines and carriage returns, and thus the input below has been reformatted a bit. The line of equals signs are the separators between the different outputs, obviously.

```

dbacaad
============================================================
aaaaaaaaaaeeeeeeeeeeeeeeeiiiiiiiiiiiisssttttt             l
============================================================
Now, naturally, in writing such a story as this, with its conditions as laid down in its
Introduction, it is not surprising that an occasional "rough spot" in composition is found
So I trust that a critical public will hold constantly in mind that I am voluntarily avoid
words containing that symbol which is, by far, of most common inclusion in writing our
Anglo-Saxon as it is, today. Many of our most common words cannot show; so I must
adopt synonyms; and so twist a thought around as to say what I wish with as much
clarity as I can.) So, now to go on with this odd contraption…
============================================================
With the help of Rolls-Royce, University Engineering and Commerce students will enjoy new
opportunities from a partnership that is part of the company's statewide investment in the
industry, Gov. Tim Kaine announced last week.

```

Pre-lab output

The output is what is listed in the encoded?.txt files. Note that the compression ratios and Huffman tree costs listed below are correct, where as the numbers listed in the encoded?.txt files are not (the values in encoded3.txt and encoded4.txt do not match).

1. normal1.txt, which when encoded, should yield encoded1.txt
    - Compression ratio of 4.30769, and Huffman tree cost of 1.85714
2. normal2.txt, which when encoded, should yield encoded2.txt
    - Compression ratio of 3.17808, and Huffman tree cost of 2.51724
3. normal3.txt, which when encoded, should yield encoded3.txt
    - Compression ratio of 1.91079, and Huffman tree cost of 4.18676
4. normal4.txt, which when encoded, should yield encoded4.txt
    - Compression ratio of 1.82064, and Huffman tree cost of 4.39407

In-lab

Out of 10 points. This lab part **is** expected to compile.

- 10 points: Huffman decoding
  - Full credit if they got the four test cases working
  - 8 points if they got it mostly right, but a few errors in the output produced. Note that the output still needs to be pretty close.
  - 3 points for writing code that compiles, but does not produce anything close to the correct output
  - 0 points for no submission or code that doesn't compile
- If they did not build the Huffman tree, they get a 0 for the in-lab

In-lab input

The four input files provided were passed as input to the submitted in-lab code, but there were eight execution runs (see the output section, below).

1. encoded1.txt, which when decoded, should yield normal1.txt
2. encoded2.txt, which when decoded, should yield normal2.txt
3. encoded3.txt, which when decoded, should yield normal3.txt
4. encoded4.txt, which when decoded, should yield normal4.txt

In-lab output

The output for each of the files, shown in the normal?.txt files, is listed below. Note that we ignore newlines and carriage returns, and thus the output below has been reformatted a bit. The line of equals signs are the separators between the different outputs, obviously.

There were eight execution runs, however, and not four. The encoded?.txt files were provided on execution runs 1, 3, 5, and 7. On the even numbered execution runs, the output of the previous one was diff'ed against the expected output. Thus, exeuction run 2 compares the output from execution run 1 (dbacaad) against the normal1.txt file (which also contains dbacaad). If they compare exactly, then no output will be produced, which is what is desired. Note that white space is ignored in this comparison.

```
dbacaad
============================================================
aaaaaaaaaaeeeeeeeeeeeeeeeiiiiiiiiiiiissstttt             l
============================================================
Now, naturally, in writing such a story as this, with its conditions as laid down in its
Introduction, it is not surprising that an occasional "rough spot" in composition is found
So I trust that a critical public will hold constantly in mind that I am voluntarily avoid
words containing that symbol which is, by far, of most common inclusion in writing our
Anglo-Saxon as it is, today. Many of our most common words cannot show; so I must
adopt synonyms; and so twist a thought around as to say what I wish with as much
clarity as I can.) So, now to go on with this odd contraption…
============================================================
With the help of Rolls-Royce, University Engineering and Commerce students will enjoy new
opportunities from a partnership that is part of the company's statewide investment in the
industry, Gov. Tim Kaine announced last week.
```

Post-lab

Out of 10 points. This lab part is expected to compile.

- 2.5 points: implementation description
  - On a scale of 0-2.5 for their quality of description of the data structures that they used, and why they selected them (half points allowed)
- 2.5 points: efficiency analysis
  - 1.5 points for the quality of their description (and explanation) of the running time of each of their encoding/decoding steps
    - 0-1.5 points, depending on the quality of this section
  - 1 point for the quality of their description (and explanation) of the worst-case space complexity for their implementation
    - 0-1 points, depending on the quality of this section
- 5 points: Objective C program
  - 5 points: if they got it correct
  - 2.5 points: if they had the right idea, but it did not work correctly (didn't print it out, etc.)

- 0 points: for no submission, or if Objective C class/methods were not used

Be the first to comment

Untitled note

Find a notebook

Add tag

Add comments

Version 6.0.6: c128369/1.0.1.250

Web Clipper tutorial

Options

To:

Saving clip...

Clip

Share

Article

Simplified Article

Save

Selection

PDF