ECS 60 Programming Assignment #4, 50 points (2.5 hours coding) Spring 2015

Due: Wednesday, May 20th, at 11:59pm to p4 directory Executable name: encoder.out
Minimum files to be submitted: Makefile, decoder.h, decoder.cpp, encoder.h, encoder.cpp, authors.txt

You are to write an Encoder class that encodes a message using lossless compression. Your Decoder class must be able to accept a message encoded by your Encoder class, and decode it into the original message. You will find encoderRunner.cpp, CPUTimer.h, a basic Makefile, my encoder.out, my encoderGold.out, data files and barebones decoder and encoder files in ~ssdavis/60/p4.

Here are the specifications:

1. Messages can be any length, but will not be previous compressed.
2. Your Decoder and Encoder classes must be completely independent of each other. Among other things, they may not share data, including global variables.
3. You may not alter encoderRunner.cpp, nor CPUTimer.h.
4. Grading
   - 4.1. Performance will be tested with large (> 500KB) examples of three types of data files: ASCI, BMP, and a 32-bit executable.
   - 4.2. I will copy encoderRunner.cpp, and CPUTimer.h into your handin directory, and then call make, so please make sure you handin all necessary files. Students often forget to handin dsexceptions.h, as well as other secondary Weiss files. "handin cs60 p4 *.cpp *.h Makefile" would probably be wise after a successful clean remake.
   - 4.3. Minimum Accuracy and Size (20 points) If your program has a encode message at least 5% smaller than the original, and provides accurate decoding, then it will earn 20 points for proper operation. If your program fails either of these requirements on any file, then you will not receive any points for the assignment.
   - 4.4. Compressed Size (15 points) min (20, 15 * Sean's Total Size / Your Total Size).
   - 4.5. CPU Time (15 points): min (20, 15 * Sean's Total CPU Time / Your Total CPU Time).
     - 4.5.1. Your program will be judged against encoder.out. CPU time may not exceed 100 for one file.
     - 4.5.2. Programs must be compiled without any optimization options. You may not use any precompiled code, including the STL and assembly. Other than Weiss files, you must have written all of the code you submit.
     - 4.5.3. You may not have any static, or global variables since they would be created before the CPU timer begins, and could be shared between your two classes. Note that you may have constants, and may load static information from your own data files in your constructors.
5. Suggestions
   - 5.1. Keep things simple, and get things running first, and only then use gprof to learn where things are going slowly.
   - 5.2. Use Weiss code where possible. His files are good starting points for any ADT you wish to use.
   - 5.3. Remember to turn in dsexceptions.h if your program needs it!
   - 5.4. If you find you have a bug in the middle of running through a message, add a an if statement in your code that describes the state of the machine at the time of the bug, and then put a breakpoint at the cout in the if statement. For example: if(index == 978 && pos == 4) cout << "Help!\n";
   - 5.5. I used Huffman encoding as the basis for my solution.
     - 5.5.1. Encodings can be more than 16 bits. Two possible options is to store the encoding in an unsigned int, or in a char array. Both have their drawbacks and advantages. You should be aware that Intel use little endian storage of ints, i.e., less significant bytes of an int have lower RAM addresses, so that 0x12345678 would be in RAM (from lowest address up) as 0x78, 0x56, 0x34, 0x12. Thus, if you use a binary write to the encodedMessage array, the four bytes will be in reverse order.
     - 5.5.2. While you will need to use bitwise shifts, ands, and ors somewhere in your program, there are efficiencies to be gained that I did not use in the benchmark.
     - 5.5.3. To encode and decode, I kept track of both the index and the position within the current char.
     - 5.5.4. While it took me only 2.5 hours to write, please be warned that it involved a wide range of programming techniques including recursive tree traversals and tree constructions.
   - 5.6. A handy piece of code to print out the bits of a message is:

```
for(int j = 0x80; j > 0; j >>= 1)
    if(encodedMessage[pos] & j)
      cout << '1';
    else
      cout << '0';
```

```
int main(int argc, char** argv)
{
  unsigned char *message, *encodedMessage, *decodedMessage;
  int encodedSize, decodedSize, size, i;
  Encoder *encoder;
  Decoder *decoder;
  struct stat filestatus;
  stat(argv[1], &filestatus);
  size = filestatus.st_size;
  message = new unsigned char[size];
  encodedMessage = new unsigned char[size];
  decodedMessage = new unsigned char[2 * size];
  ifstream inf(argv[1], ios::binary);
  inf.read((char*) message, size);
  inf.close();
  CPUTimer ct;
  encoder = new Encoder();
  encoder->encode((const unsigned char*) message, (const int) size, encodedMessage,
    &encodedSize);
  delete encoder;
  decoder = new Decoder();
  decoder->decode((const unsigned char*) encodedMessage, (const int) encodedSize,
    decodedMessage, &decodedSize);
  cout << dec << "CPU time: " << ct.cur_CPUTime() << " Encoded size: "
      << encodedSize;

  for(i = 0; i < filestatus.st_size && message[i] == decodedMessage[i]; i++);

  if(i == filestatus.st_size && decodedSize == filestatus.st_size)
    cout << " OK\n";
  else
    if( message[i] != decodedMessage[i])
      cout << "\nMismatch at location " << i << hex << showbase <<" original: "
        << (int) message[i] << " decoded: " << (int) decodedMessage[i] << endl;
    else
      cout << "\nMismatch of sizes, original: " << filestatus.st_size
        << " decoded: " << decodedSize << endl;

  return 0;
} // main()

[ssdavis@lect1 p4]$ ls -l extract4.bmp
-rw-r--r-- 1 ssdavis users 529974 May  9  2013 extract4.bmp
[ssdavis@lect1 p4]$ encoder.out extract4.bmp
CPU time: 0.035289 Encoded size: 226450 OK
[ssdavis@lect1 p4]$ encoderGold.out extract4.bmp
CPU time: 0.033424 Encoded size: 226707 OK
 [ssdavis@lect1 p4]$ ls -l GlobalAirportDatabase.txt
-rw-r--r-- 1 ssdavis users 559104 May  3 11:01 GlobalAirportDatabase.txt
[ssdavis@lect1 p4]$ encoder.out GlobalAirportDatabase.txt
CPU time: 0.057727 Encoded size: 305962 OK
[ssdavis@lect1 p4]$ encoderGold.out GlobalAirportDatabase.txt
CPU time: 0.054845 Encoded size: 306219 OK
[ssdavis@lect1 p4]$ ls -l aside.out
-rwx------ 1 ssdavis users 4087555 May  3 11:07 aside.out
[ssdavis@lect1 p4]$ encoder.out aside.out
CPU time: 0.552599 Encoded size: 3123904 OK
[ssdavis@lect1 p4]$ encoderGold.out aside.out
CPU time: 0.515264 Encoded size: 3124161 OK
[ssdavis@lect1 p4]$
```