Assignment 6

Assignment 6 is due at **5 pm Friday, December 1**. It is worth 35 points.

Procedures

Turn in answers to the exercises below on the UAF Blackboard Learn site, under Assignment 6 for this class.

- Your answers should consist of two files: `huffcode.hpp` and `huffcode.cpp`, from Exercise A. These two files should be **attached** to your submission.
- Send only the above! I do not want things I already have, like the test program.
- I may not look at your homework submission immediately. If you have questions, e-mail me.

Exercises (35 pts total)

Exercise A — Huffman Codes

Purpose

In this exercise, you will implement Huffman coding. Given character weights, your package will be able to construct a Huffman code, which it can then use to encode and decode strings.

Instructions

Write a class `HuffCode`, defined in files `huffcode.hpp` and `huffcode.cpp`.

Class `HuffCode` should include the following public member functions.

- Default constructor, copy constructor, copy assignment, destructor. All of these should do the usual things.
  - So you can use the compiled-generated versions, as long as you do not do anything silly like declaring a data member that is a pointer.
- Member function `setWeights`, prototyped as follows (remember, this and later functions are all class members):

  ```
  void setWeights(const std::unordered_map<char, int> & theweights);
  ```

  Function `setWeights` sets the character weights. It is given an `unordered_map`, as above; the value corresponding to a

character is its weight. Characters whose weight is not specified will not occur in any texts to be encoded.

- Member function `encode`, prototyped as follows:

  ```
  std::string encode(const std::string & text) const;
  ```

  Given a string of characters, each of which has already had its weight defined (using `setWeights`), this returns a string of zero (`'0'`) and one (`'1'`) characters, representing the given text, encoded using an appropritate Huffman code. *See Examples, below, for an example.*

- Member function `decode`, prototyped as follows:

  ```
  std::string decode(const std::string & codestr) const;
  ```

  Given a string of zeroes and ones, encoded using the Huffman code generated by the class, this returns the corresponding text. In particular, if the return value of `encode` is passed to this function, then it will return the argument of `encode`. *See Examples, below, for an example.*

Your code may generate and use any Huffman code that is correct for the given weights.

Examples

```
HuffCode h;
std::unordered_map<char, int> w;
w['a'] = 1;
w['b'] = 2;
w['c'] = 3;

h.setWeights(w);
// I will assume the code is
//   a: 00
//   b: 01
//   c: 1
// There are other possible Huffman codes for the above weights;
//  these would result in different behavior below.
std::cout << h.encode("aca") << std::endl;
// Above prints "00100"
std::cout << h.decode("00100") << std::endl;
// Above prints "aca"
std::cout << h.decode("01000111") << std::endl;
// Above prints "babcc"
```

Skeleton Files

I have provided unfinished "skeleton" files `huffcode.hpp` and `huffcode.cpp`. You may use these as the basis for your own work, if you wish.

Test Program

A test program is available: `huffcode_test.cpp`. If you compile and run this program (unmodified!) with your code, then it will test whether your code works properly.

Do not turn in the test program.

Notes

- Coding standards are as for Assignment 2.
- Your code may be tested with additional input beyond that given in the posted test programs.