School of Computing & Information Technology

CSCI251/CSCI851 Advanced Programming Spring 2019

Assignment 3 (Worth 10%)

Due 11:55pm Saturday 2^{nd} November 2019. (End of Week Thirteen)

This assignment involves implementing container and class functionality to support measuring the properties of certain classes of codes.

General notes

These are some general rules about what you should and shouldn't do.

1. Your assignment should be sensibly organised with the same kind of expectations in this area as the previous assignments. No memory leaks etc. too.
2. Your code must compile on Banshee with the compilation instructions you provide in `Readme.txt`. If it doesn't you will likely receive zero for the coding part of the assignment. Note that you will have to use `-std=c++11` for the inclusion to work correctly.
3. **Other than the initial command line input, the program should run without user input.**
4. You should use classes.
5. You should use templating.
6. You shouldn't use inheritance.
7. Failure to comply with any of the three points above will result in significant penalties.

Some coding (theory) containers

You are to implement two containers:

1. A `Codeword container` for storing ordered lists of symbols making up a codeword.
2. A `Codebook container` containing a collection of codewords.

The symbols are themselves allowed to be of any type that supports certain functionality. We shall consider the containers after considering the two symbol classes you need to implement.

The symbol classes

Note that you should not relate these classes by inheritance. Each instance of these classes stores a single value.

1. The first symbol class `Mint` allows us to store a single integer value in the range 0 to $p-1$, where p is a positive integer, the modulus. In this class you should overload the subtraction operator overloaded to produce the typical integer result modulo p.

2. The second symbol class `Melt` allows us to store a single lowercase letter from the English alphabet, with the subtraction operator overloaded to give 1 if the symbols are different and 0 if they are the same. There are $p = 26$ distinct possible symbols.

See later for a description of the `main()` function. Both should also have a special "zero" symbol, for `Mint` this will be the integer 0, while for `Melt` it will be a letter 'a'.

The `Codeword` container

This container should be used to store elements of the same type. It should be a templated container class. The following methods should also be provided.

1. The method `Weight` should determine the number of elements in the code that are not–equal to the "zero" symbol. For example, `0 2 0 3 3 3 0` has a weight of 4.

2. The method `Distance` should take another `codeword` and determine the sum of the element by element difference according to the overloaded subtraction operation for the contained symbol class.

   For example, the distance between `0 1 2 3 0 1` and `0 0 2 0 0 2` for a `Mint` container with $p = 7$ would be determined as

   ```
     0 1 2 3 0 1
   - 0 0 2 0 0 2
   --------------
     0+1+0+3+0+6 = 10
   ```

3. The method `Display` should output the elements in the codeword, each separated by a space, with a final gap and the weight of the codeword displayed. For example,

   ```
   4 6 9 11 3     Weight: 5
   ```

The `Codebook` container

This container should be used to store collections of `codeword`s. It should be a templated container class. The following methods should be provided.

1. The method `minimumWeight` should determine the minimum `Weight` value across all non–zero codewords.

2. The method `calcDistance` should determine the distances between every pair of codewords in the code, and store these values.

3. The method `minimumDistance` should determine the minimum `Distance` between two codewords in the code, as determined across all distinct pairs of codewords.

4. The method `Display` should display all the codewords contained in the container, using the `Display` method for the codewords themselves, and display the minimum weight and minimum distance for this code. The table of distances between codewords should be displayed also.

This container should contain a zero codeword, with the elements being all "zero", in accordance with the zero element for the contained symbol class.

Containers for Codes: The `main()` function

Once your program is compiled into the executable `CFC` it must run as follows:

./CFC 0 seed length size modulus

./CFC 1 seed length size

where the parameters have the following meanings:

- `First argument` : (0 for `Mint`, 1 for `Melt`).
- `seed` : A positive integer. Random seed for use in the value generator functions.
- `length` : A positive integer. The number of symbols in each codeword.
- `size` : A positive integer. The number of codewords in each codebook.
- `modulus` : A positive integer, only relevant for the `Mint` code.

You generate values to populate your codewords by making calls to the functions provided in `generateValue.h` and `libGenVal.a`, using seed and modulus as arguments as needed.

After populating the codebook, you should calculate the minimum weights and distances for the code. The code should be displayed using the code method `Display()`.

Some test data

Here goes a couple of test data cases. These are the actual codebooks, including the the zero codewords. See also some examples in the Week 11 lecture/tutorial.

```
$ ./CFC 0 100 5 4 23
0 0 0 0 0
12 15 6 9 9
12 19 3 0 3
2 4 15 4 18

$ ./CFC 1 101 5 4
a a a a a
n o o y a
z e a r n
v h h p x
```

Notes on submission

Submission is via Moodle.

Your code must compile on Banshee with the instructions you provide. If it doesn't you will likely be given zero for this assignment.

Please submit your source, so .cpp and .h files, and Readme.txt file, and makefile if you have one, directly to Moodle. There shouldn't be other files or directories and they shouldn't be in a zip file. Please don't submit `generateValue.h` or `libGenVal.a`. Submission is via Moodle.

1. The deadline is 11:55pm Saturday 2^{nd} November 2019.
2. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
3. Submissions more than three days late will not be marked, unless an extension has been granted.
4. If you need an extension apply through SOLS, if possible **before** the assignment deadline.
5. Plagiarism is treated seriously. Students involved will likely receive zero.