

COSC 3360 - Fundamentals of Operating Systems

[Dashboard](#) / [My courses](#) / [COSC3360SP2022](#) / [PROGRAMMING ASSIGNMENTS](#) / [Programming Assignment 1](#)


 [Description](#)

 [Submission](#)

 [Edit](#)

 [Submission view](#)

Programming Assignment 1

 **Due date:** Sunday, 20 February 2022, 11:59 PM

 **Requested files:** main.cpp ( [Download](#))

Type of work:  Individual work

Similarity Threshold: 90%

Problem:

For this assignment, you will create a parallel fixed-length code decompressor using the tools we learned in class to create multiple processes and threads.

Computers represent a character as a sequence of 8 bits, which means you can represent up to 256 symbols per character. However, depending on the source, it is common to have messages/files using an alphabet with a size of fewer than 256 symbols.

The simplest way to achieve compression is to use a fixed-length code generator. The idea behind this type of code generator is to assign a fixed-length bit sequence to each alphabet symbol.

Given an alphabet (where each symbol is represented by a character and a decimal value with the code of the symbol), you need to implement a fixed-length code decompressor based on the following steps:

- Read the contents of the input file using input redirection (STDIN). The format of the input file is as follow:
 - An integer value representing the number of symbols in the alphabet
 - n lines (where n is the number of symbols in the alphabet) with a char representing the value of the symbol and an integer value representing the symbol's code (in decimal notation).
 - A string representing the compressed message (sequence of bits).

Given the previous format, the following file represents a valid input file:

```
3
a 2
b 4
c 5
010010010100101101101
```

- Calculate the number of bits of the fixed-length codes based on the value of the greatest base 10 code in the alphabet. To calculate the number of bits used per symbol, you will use the following formula: $\text{ceil}(\log_2(\text{max code value in the alphabet}))$.
- Create n child processes or threads (where n is the number of symbols in the alphabet). Each child process or thread executes the following tasks:
 - Determines the binary representation of the symbol's code.
 - Determines the frequency of the symbol in the compressed message.
- Create m child processes or threads (where m is the number of characters in the decompressed message) to determine each character of the decompressed message.

- Print the information about the alphabet and the decompressed message. Given the previous input file, the expected output is:

Alphabet:

Character: a, Code: 010, Frequency: 3

Character: b, Code: 100, Frequency: 1

Character: c, Code: 101, Frequency: 3

Decompressed message: aaabccc

NOTES:

- You must select the appropriate tool to implement the parallel fixed-length code decompressor. Not using multiple threads or multiple processes will translate into a penalty of 100%.
- You must use the output statement format based on the example above. Only printable characters will be considered valid input to guarantee consistency when printing the alphabet's characters to STDOUT.
- You can safely assume that the input will always be valid.

Requested files

main.cpp

```
1 // Write your program here
```

[VPL](#)

[← Announcements](#)

Jump to...

[Using Virtual Programming Lab \(VPL\) →](#)