CS 2413 – Data Structures – Fall 2024 – Project Five
Due 11:59 PM, November 25, 2024
(no Penalty Until November 30)

Description: Encoding is a process of converting data from one form to another. Encoding in many cases reduces the number of bits required to store the original file, in which case it is called compression. In this project, you will do one such encoding, where you read a text file and replace each word with a number.

Consider the following text (John F. Kennedy – Inauguration Address):

> Can we forge against these enemies a grand and global alliance, North and South, East and West, that can assure a more fruitful life for all mankind? Will you join in that historic effort? In the long history of the world, only a few generations have been granted the role of defending freedom in its hour of maximum danger. I do not shrink from this responsibility. I welcome it. I do not believe that any of us would exchange places with any other people or any other generation. The energy, the faith, the devotion which we bring to this endeavor will light our country and all who serve it and the glow from that fire can truly light the world. And so, my fellow Americans: ask not what your country can do for you; ask what you can do for your country. My fellow citizens of the world: ask not what America will do for you, but what together we can do for the freedom of man.

A token is a string of characters that is separated by a space or end of line character. For example, mankind? (with the question mark) is a token. The token "danger." (with the period at the end is a token). The token "The" and "the" are two different tokens.

The goal of this project is to get all tokens from an input file and then count the number of times each token occurs. For example, in the above text, the token "the" occurs 4 times. Once you have the tokens and their frequencies, you need to sort the tokens in the decreasing order of their frequencies. We then output, all the unique tokens, separated by a space, followed by a number of integers again separated by a space that represents the position of each token in the text in the sorted set. For example, if the word "the" is the second token (in the index position 1) in the sorted set of tokens, we will replace the token "the" in the text with the number 1, and so on.

The learning objectives for this project include:

1) Parsing the text file and obtaining tokens (learn input/output to files)
2) Learning to use the multimap data structure that is part of the C++ standard library
3) Learning how encoding works

Here is a simple code segment (you need to check and fix if any errors) that read tokens from a file:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {

    ifstream myInputFile ("input1.txt");
    string token;
    char aChar;

    while (!myInputFile.eof()) {
            do {
                    myInputFile.get(aChar);
            } while ((!myInputFile.eof()) && ((aChar == ' ') || (aChar == '\n')));

            if (!myInputFile.eof()) {
              token.clear();
              while ((!myInputFile.eof()) && ((aChar != ' ') && (aChar != '\n'))){
                    token += aChar;
                    myInputFile.get(aChar);
               }
              cout << token << endl;
            } //end if
           } //end while
           return 0;
}
```

Once you have mastered the above program, you will understand the map and unordered map data structure in C++ (simply google it). Your project will have these additional includes as well.

```
#include <unordered_map>
#include <map>
#include <iterator>
#include <algorithm>
```

You will see that the unordered map data structure can be used to store string object (unordered map will use this as the KEY) and its frequency of occurrence (unordered map will use this as the VALUE). Once you store all the tokens and their frequencies, you need to sort (you can use the sort() in the algorithm include – google it too), the tokens in the descending order of their frequencies.

The outline of the entire program will be something like this:

```
#include <iostream>
#include <fstream>
#include <string>
#include <unordered_map>
#include <map>
#include <iterator>
#include <algorithm>

using namespace std;

int main () {

  //read the name of the file ./a.out < filename.txt

  //get each token and store them in the unordered_map (or map) increment
  //its frequencies. You MAY be able to do myTokens[aToken]++. Work on this.

  //close the file (filename.txt)
  //sort the myTokens in the decreasing order of VALUE which is frequencies
  //print the KEYS in myTokens (which has been sorted)separated by a space.

  //after you printed the KEYS Do this
  cout << endl;
  cout << "********" << endl;

  //Now open the filename.text file again for reading
  //Read token by token as you have done this before
  //each time you get a token, find its position in the myTokens (sorted
  //data structure and print the position followed by space

  cout << endl;

  return 0;
}
```

FOR BONUS CODE:

You must submit separate project5_decompress.cpp file. You will be having two links for submission on canvas, one for common project-5's code and one for bonus code. You are allowed to reuse above logic for bonus part. We won't be counting any bonus if you submit your bonus code in project5.cpp. If you do so, the gradescope test case will fail and you won't get points for that. So, make sure you submit it separately.

BONUS: For **30% extra credit**, write another program, name your program as project5_decompress.cpp that takes the output of the above program and produces text.

Constraints

1. You are allowed to use only the libraries given above.
2. You need to work individually in this project.
3. Any use of internet's resources need to be cited in your code.

Project Submission Requirements:

1. **Code Development (75%):** Implement the provided classes and the required methods Your implementation must be fully compatible with the main program provided. Your code must run successfully with the main program and the corresponding input, demonstrating the correct functionality of all classes and its methods.
2. **LLM/AI Tool Usage:** You can use Large Language Models (LLMs) or AI tools, such as GitHub Copilot, to assist in writing and refining your classes and their methods. If you did not use LLM or AI tools to write your project, you still have to show for 2. below how you would have used it to find a solution to the project.
3. **LLM and GitHub Copilot Usage Documentation (15%):** If you choose to use LLM tools or GitHub Copilot, you must document your usage. This documentation (in PDF Format) should include:

    - **Prompts and Suggestions:** Provide the specific prompts or suggestions you used, such as "Generate a method for this method"
    - **Rationale:** Explain why you chose these prompts or suggestions and how they contributed to the development of your classes. For instance, you might describe how a particular suggestion helped you structure the program.
    - **Incremental Development:** Detail how you used the tools to build and refine your classes and methods incrementally. For example, you might start by generating a basic structure for the classes and then refine individual methods, ensuring each one integrates smoothly with the main program.

4. **Debugging and Testing Plan (10%):** Submit a comprehensive debugging and testing plan. This should include:

    - **Specific Tests:** Describe the tests you conducted on your class methods.
    - **Issues and Resolutions:** Document any issues you encountered, such as handling zero values or optimizing for performance, and how you resolved them.
    - **Verification:** Explain how you verified that your classes work correctly with the provided main program. This could involve running a series of test cases the main program provides or creating additional test cases to ensure robustness.