Laiba Maab

CID: DEP2248

Task3: Implementing a Simple File Compression Algorithm

```
#include <iostream>
#include <fstream>
#include <string>
std::string readFile(const std::string& fileName);
void writeFile(const std::string& fileName, const
std::string& data);
std::string compressRLE(const std::string& data);
std::string decompressRLE(const std::string& data);
int main()
{
   std::string inputFileName = "input.text";
   std::string compressedFileName = "compressed.rle";
   std::string decompressedFileName =
"decompressed.text";

   std::string inputData = readFile(inputFileName);
   if (inputData.empty())
   {
      std::cerr << "Input file is empty or could not be
read." << std::endl;
      return 1;
   }
```

```
    std::string compressedData =
compressRLE(inputData);
    writeFile(compressedFileName, compressedData);


    std::string decompressedData =
decompressRLE(compressedData);
    writeFile(decompressedFileName, decompressedData);


    std::cout << "Compression and Decompression
completed." << std::endl;
    return 0;
}


std::string readFile(const std::string& fileName)
{
    std::ifstream file(fileName, std::ios::binary);
    if (!file.is_open())
    {
        std::cerr << "Cannot open the file " << fileName <<
std::endl;
        return "";
    }
    std::string
content((std::istreambuf_iterator<char>(file)),
std::istreambuf_iterator<char>());
    file.close();
    return content;
```

```
}


void writeFile(const std::string& fileName, const
std::string& data)
{
   std::ofstream file(fileName, std::ios::binary);
   if (!file.is_open())
   {
      std::cerr << "Cannot write to the file " << fileName
<< std::endl;
      return;
   }
   file.write(data.c_str(), data.size());
   file.close();
}


std::string compressRLE(const std::string& data)
{
   std::string compressed;
   int n = data.length();
   for (int i = 0; i < n; ++i)
   {
      int count = 1;
      while (i < n - 1 && data[i] == data[i + 1])
      {
         ++i;
```

```
            ++count;
        }
        compressed += data[i];
        compressed += std::to_string(count);
    }
    return compressed;
}

std::string decompressRLE(const std::string& data)
{
    std::string decompressed;
    int n = data.length();
    for (int i = 0; i < n; ++i)
    {
        char ch = data[i];
        std::string countStr;
        while (i + 1 < n && std::isdigit(data[i + 1]))
        { countStr += data[++i]; }
        int count = std::stoi(countStr);
        decompressed.append(count, ch);
    }
    return decompressed;}
```

https://onlinegdb.com/7TTXq-f_v