#include <iostream>  
#include <fstream>  
#include <string>  
#include <map>  
#include <queue>  
#include <math.h>  
using namespace std;  
int main() {  
 char c; //the character being read from the file  
 long i = 0; //tracks number of characters read in to trigger window  
 float cgPercent = 0; //the percent of C and G within the window  
  
 map<char, int> charCounter; //map used to count nucleotide frequency  
 queue<char> window; // the sliding window of nucleotides that is being examined for C and G  
  
 ifstream geneFile("C:/Users/jeffp/CLionProjects/HPC1/Human\_chromosome-1\_PARK7.fasta"); //input file stream  
 ofstream output("C:/Users/jeffp/CLionProjects/HPC1/output\_park7.txt"); //output file stream  
  
 output << "C and G combined frequency > 75% windows:\n"; //write a title for the window table  
  
 if (geneFile.is\_open()) { //confirm file is open  
 while (geneFile >> c) { //read in characters, skipping whitespace  
 charCounter[c]++; //increment the character count in the map  
 window.push(c); //add the character to the window  
 i++; //increment number of characters  
 if (c == 'C' || c == 'G') { //if the character is C or G, increment the CG percentage tracker  
 cgPercent += 0.2; //since 1/500 = 0.2%, add that  
 }  
 if (i > 500) { //once window has filled, begin removing elements at the head  
 if (window.front() == 'C' || window.front() == 'G') { //decrement CG percentage tracker if needed  
 cgPercent -= 0.2;  
 }  
 window.pop();  
 }  
 if (cgPercent > 75) { //if the window contains > 75% CG, write it to output  
 output << i - 500 << ", " << ceilf(cgPercent \* 100) / 100 << "\n";  
 }  
 }  
 geneFile.close();  
 }  
 else {  
 cout << "Unable to open file";  
 }  
  
 //calculate the frequency of each nucleotide, excluding unknowns, and write to output  
 float totalNuc = charCounter['A'] + charCounter['T'] + charCounter['C'] + charCounter['G'];  
  
 float aFreq = (float)charCounter['A'] / totalNuc \* 100;  
 float tFreq = (float)charCounter['T'] / totalNuc \* 100;  
 float cFreq = (float)charCounter['C'] / totalNuc \* 100;  
 float gFreq = (float)charCounter['G'] / totalNuc \* 100;  
  
 output << "\nFrequencies without unidentified:\n";  
 output << "A: " << aFreq << "%\n" << "T: " << tFreq << "%\n" << "C: " << cFreq << "%\n" << "G: " << gFreq << "%\n\n";  
  
 //calculate the frequency of each nucleotide, including unknowns, and write to output  
 totalNuc += charCounter['N'];  
  
 aFreq = (float)charCounter['A'] / totalNuc \* 100;  
 tFreq = (float)charCounter['T'] / totalNuc \* 100;  
 cFreq = (float)charCounter['C'] / totalNuc \* 100;  
 gFreq = (float)charCounter['G'] / totalNuc \* 100;  
 float nFreq = (float)charCounter['N'] / totalNuc \* 100;  
  
 output << "Frequencies with unidentified:\n";  
 output << "A: " << aFreq << "%\n" << "T: " << tFreq << "%\n" << "C: " << cFreq << "%\n" << "G: " << gFreq << "%\n" << "N: " << nFreq << "%\n";  
 output.close();  
  
 return 0;  
}