MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 4: A Simple Email Spam Analyser

This assignment aims to help you practice ***string processing*** and ***string matching.*** Your main task in this assignment is to develop a simple email spam analyser using **C programming language**.

Overview

Everyone who has ever used an email has experienced spam and phishing emails. Whether it's an email claiming to be your bank telling you to send your credit card info, or The National Lottery telling you about your win and how you only need to click the link below to claim your prize.

Interestingly, these spam emails have certain patterns and commonly occurring words that allow for quick detection, which why email clients can sometimes automatically label an email as spam. In this assignment, you will need to develop a simple email analyser to check whether a given pattern is available in a set of non-spam and spam emails.

This application will start by taking the path of a directory in which a bunch of text files are stored. Each text file has an email stored in it in the following structure.

Email_ID
From: Sender
To: Recipient
Date: Day_of_the_month
Type: Spam/Nonspam
Content

As you can notice, there is an extra line in the email. If it "Spam", then the email is a spam email, otherwise it's not. The program will read each of these emails into an appropriate structure. The email structure will look like:

Information	Data Type
Email_ID	int
Sender	char[51]
Recipient	char[51]
Day_of_the_month	int
Spam/Nonspam	char[8]
Content	char*

This application will then ask the user to enter the pattern to be detected (such as, "won"). It will then show the spam or non-spam emails that include the pattern by highlighting the patterns with square brackets. It will also show how many pattern matches are found in how many emails. If there are multiple of them, all of them should be printed. A sample run is provided below:

```
Enter the directory path containing the emails: C:\Users\user\Desktop\data4
Enter the number of emails to be read: 8
Enter the pattern: won

Spam emails containing the pattern:
1. We would like to inform you that you have [won] a grand prize of
$1,000,000. Click the link below to claim the money you [won]!
2. We would like to inform you that you have [won] an Iphone 14! Click the
link below!
3 pattern(s) detected in 2 email(s)
```

```
Non-spam emails containing the pattern:
1. You know I [won] that bet, send me the money.
1 pattern(s) detected in 1 email(s)
```

Please note that the search for the pattern should be case insensitive, meaning the if the program is search for the pattern "won" and the word "WON" occurs in the email, it will still detect the pattern.

Implementation Requirements

This application will have one linked list for spam emails and another one for non-spam ones. In each node of the linked lists, an email structure should be kept which has the information of the corresponding email. You will need to implement the following functions and call them appropriately in the main function. You are suggested to implement helper functions to make your program more modular.

- **readEmails**: This function takes the path to the directory which contains the data files, the number of data files, and two empty linked lists (one for spam emails and another one for non-spam emails). It then reads the files that are named according to a number. For example, if there are 50 data files, then the files will be named "1.txt", "2.txt", all the way to "50.txt". When it reads an email, it will create a node for it and add it to its appropriate linked list.

  Be sure to add a few error checks to this function, such as when the user enters a path that does not exist or a number of emails that is more than the one found in the entered directory. Moreover, note that there is increased restriction on this function being implemented correctly because if it is not, this will mean that the rest of the code will not be testable, and you will consequently lose a significant amount of points for other parts.

- **searchPattern:** This function takes the pattern and the linked lists of spam and non-spam emails to check whether the pattern is available in the emails by using the **Rabin-Karp Algorithm** where d=128 (since there are 128 alphanumeric) and q=256. If the pattern is detected in an email, it will be highlighted with square brackets as shown above. The general pseudo-code of the **Rabin-Karp Algorithm** is given below.

```
RK(T, P, d, q)
      n := length[T];
      m := length[P];
      h := d^(m-1) mod q;
      p := 0;
      t_0 := 0;
      for i := 1 to m do
            p := (dp + P[i]) mod q;
            t_0 := (dt_0 + T[i]) mod q
      for s := 0 to n – m do
            if p = t_s then
                  if P[1..m] = T[s+1..s+m] then
                        print "pattern occurs with shift s"
            if s < n–m then
                  t_s+1 := (d(t_s – T[s+1]h) + T[s+m+1]) mod q
```

As mentioned above, this function will also show how many pattern matches are found in how many emails for each group of emails.If there are multiple of them.

Incremental and Modular Development

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions.

Professionalism and Ethics

You are expected to complete the assignments on your own. Sharing your work with others, uploading the assignment to online websites to seek solutions, and/or presenting someone else's work as your own work will be considered as cheating.

Rules

- You need to write your program by using **C programming language.**
- You need to create your own data files for testing your program.
- You need to name your file with **your student id**, e.g. 1234567.c, and submit it to ODTUCLASS.
- **Code quality, modularity, efficiency, maintainability, and appropriate comments** will be part of the grading.

Grading Policy

The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Data Representation	5
Main Function & Appropriate Function calls in the Entire Program	10
readMails	20
searchPattern - Use of Rabin-Karp Algorithm	25
searchPattern - Print scanpaths that includes the pattern	20
searchPattern – Print the statistics (the number of pattern matches, the number of emails include the pattern)	20