ECE 404 Homework #11

Due: Tuesday 4/11/2023 at 5:59PM

Spam Filter

Design spam filter recipes that will trap all 74 messages that you will find in the gzipped tar archive **junkMail.tar.gz** uploaded on Brightspace along with the assignment. When you gunzip and untar the archive with, say,

```
tar -zxvf junkMail.tar.gz
```

you'll see 74 individual spam messages with names **junkMail_1** through **junkMail_74**. About these messages:

1. **junkMail 1 through junkMail 50** : The headers of all these messages have one thing in common: they contain multiple entries in the "From:" header. All these messages were trapped by a single recipe in your instructor's spam filter. The regex in your instructor's recipe has only 40 characters in it. (If the regex engine used by procmail allowed for Perl's $'\{\}'$ metacharacters, this regex could have been made as short as just 10 characters.)

2. **junkMail 51 through junkMail 63** : These messages can be trapped just on the basis of the "Subject:" line in the email headers.

3. **junkMail 64 through junkMail 66** : In your instructor's spam filter, these messages were trapped on basis of the content (email body) of the messages.

4. **junkMail 67 through junkMail 74** : You need to trap these with a single recipe that contains compound rules. Below is an example of a recipe with compound rules. It is NOT the compound recipe for trapping the messages junkMail 67 through junkMail 74:

```
:0 HB:
* ^Content-Type: text/plain
* !^Content-Type: text/html
* !^content-type: application/pdf
* !^content-type: application/zip
* !^content-type: application/msword
* !^content-type: application/.*signature
* Content-Transfer-Encoding: base64
junkMailCompound6
```

   This recipe says that if the "Content-Type" MIME header is text/plain and none of the MIME objects are of type PDF, ZIP, etc., and yet the "Content-Transfer-Encoding" MIME header calls for Base64 encoding, then there is a great chance it is a spam message.

Spam Filter Requirements

- You should have a recipe for each of the four groups described above. **Design your recipes such that each recipe does not capture junkMail intended for *later* recipes** (e.g. recipe 2 should not capture junkMail 64 through 66) .
- While it is hypothetically possible that some earlier junkMail could be caught by later recipes based on their criteria, **this should not happen since the emails are processed based on the recipe order**. For example, recipe 4 *could* catch an earlier junkMail based on its criteria, but since that junkMail would already be captured by an earlier recipe, it should not happen.
- Each recipe should **write the junkMail to one of the following files** based on the recipe number: `recipe_1`, `recipe_2`, `recipe_3`, and `recipe_4`

Useful Notes

- After you have incorporated the new recipes in your **.procmailrc** file, your filter can be tested on an individual message by invoking the command:

  ```
  procmail .procmailrc < junkMail_XX
  ```

  where "XX" is the integer suffix for the message file. Obviously, you would need to write either a shell script, or a Python script, or a Perl script to execute the above command in a loop for all 74 spam messages.
- **It is advised that you use SSH (on shay.ecn.purdue.edu) instead of ThinLinc.** In the past students have had trouble using procmail when logging in via ThinLinc, particularly when using the procmail command.
- If your recipes work on all 74 messages that have been sent to you, you will not see any messages being subject to the default action of your procmail filter, which is usually to put the surviving messages in your mailbox **/var/mail/account_name** (this can be viewed with the mailx command). Of course, you should test your recipes with your own messages that shouldn't be marked as spam (you will need to create your own emails that will specifically avoid the spam criteria for this homework). This way, you can ensure that your recipes allow the desired messages through.
- Since the spam message in the tar archive are in their raw form, it is sometimes hard to see what is in them − especially if the MIME objects in the message are Base64 encoded. To decipher those spam messages that are fully or partially encoded, you can use Prof. Kak's Perl script **EmailParser2.pl** found in Lecture 31. Execute this script (you may need to modify the shebang line based on where perl is installed for you) and give it a command-line argument that is the name of the junk mail file you want to decipher. It will deposit the different MIME objects in the email in a subdirectory called **mimemail** in the directory in which you execute the script.
- If you have trouble using the EmailParser2.pl script, you may want to install the `mutt` email client on your personal computer to help you read the emails.

Submission Instructions

- For this homework you will be submitting a zip file titled `hw11_<last name>_<first name>.zip`, which consists of:
  - A pdf titled `hw11_<last name>_<first name>.pdf` containing:
    - A brief description of how you crafted each recipe.
  - A well commented copy of the `.procmailrc` file you used to filter the junk mail.