CMPT 126 - Summer 2015

Karol Swietlicki

Assignment 2 - Simulating the global postal system
Deadline: June 14th - Sunday

Contents

1 A lesson in dealing with difficulties

The world is a complex thing. It is impossible to deal with all of the complexities, but we can create pretty good approximations. This model of the global postal system is also just a simplification of reality, but with a grain of truth to it and based on my own experiences dealing with international mail.

Getting items from point A (the initiating end) to point B (the destination end) is not as easy as it seems. Your job is to write code which will simulate the motions of letters and packages through a period of time. Many things can go wrong. Packages can be rejected before they get sent. Letters may have to be returned to sender. The postal furnaces and smokestacks are always hungry for mail which happens to be in the wrong place at the wrong time. It is not fun to be a piece of mail and even less fun to be a postal system designer. This is where computing science comes into play.

Simulations such as this one allow analysis of a system in artificial conditions that we can impose. We can provide predicted test inputs and check if, for example, our new post office would be able to handle the load that will likely be put upon it. It is cheaper to run many simulations than to build a too-small post office and have to spend a lot of money on trying to enlarge it. Or maybe the post office is too big and will never need the space that is allocated to it. The amazing thing is that years of practice and many courses of theoretical preparation are not at all a requirement for making very useful programs. This assignment requires nothing beyond file I/O, array operations, loops and the basics of abstraction.

This is a difficult assignment, mainly by the virtue of its very intimidating size. Management of task complexity is probably the most important skill you will learn throughout your studies and the sooner you learn it, the better for you. A good strategy to tackle gigantic problems is to work only on one facet of your task at a time. Once you are done with one fragment, move on to another.

To facilitate such a piecewise approach, I have have provided to you a list of requirements that I expect from the software that you are to write. If your program complies with a requirement, you will get a point for that requirement. If your program performs the task incorrectly, you don't get a point for that requirement. Your mark on the assignment will be equal to number of the tasks performed correctly out of all of the required tasks. It is very easy to earn partial marks for this assignment, but hard to get all of the points.

I will provide you with sample inputs which test all the requirements, so you can check your work and know which requirements you still need to work on. I will use almost identical inputs to grade your code, so you should be able to estimate your mark with very high confidence just by running the sample inputs against your software.

The single most challenging part of this assignment will probably be deciding how to represent simulated entities in your code. The second most difficult part will probably be dealing with the input formats. The rest of the code is pretty straightforward, if voluminous. Start work early and complete a requirement or two a day and you will be done before you know it.

There is a very large amount of duplication in the tasks you have to accomplish. This is by design, to help you spot such patterns when it is time to deal with actual coding tasks. To make your software development go faster you should write code which can handle mailed items in a generic fashion whenever possible. There are some differences between letters and packages, but often they can be treated identically and through common code. Different kinds of log messages also have a lot in common to each other and your work can be made much simpler if you focus on exploiting such similarities.

If you don't exploit the repetitious nature of this problem, your code will grow large, buggy and unmanageable. If you don't use abstraction this assignment will be almost impossible, which is exactly the point of this exercise.

2 The list of requirements

The requirements are listed in the suggested order of implementation. There are 25 requirements total. Each completed requirement contributes 4% to your assignment score. There are no partial marks for incomplete or buggy requirements.

2.1 Log file creation and day logging

1. At program startup create a master log file and a front page log file. The master log is the file called "log_master.txt" and the front page log is the file called "log_front.txt".
2. At program startup create a log file for each post office in the system. The log files are called "log_X.txt" where X is the name of the post office to which the log file belongs.
3. At the end of each simulated day write an end-of-day entry to the master log and to the log of each post office.

2.2 Arrival of items

1. When a letter is brought for mailing to a post office, write a new-letter entry to the log of the initiating post office.
2. When a package is brought for mailing to a post office, write a new-package entry to the log of the initiating post office.

2.3 Rejection of items

1. If any item has a nonexistent post office as the destination, write a rejected-package or rejected-letter entry to the log of the initiating post office and to the master log.
2. If any item's recipient is a person from the internationally wanted criminals list, write a rejected-package or rejected-letter entry to the log of the initiating post office and to the master log.
3. If a package has insufficient funds to cover the postage, write a rejected-package entry to the log of the initiating post office and to the master log.
4. If a package has sufficient funds to cover the postage, but exceeds the length restrictions of the initiating post office, write a package-rejected entry to the log of the initiating post office and to the master log.
5. If a package has sufficient funds to cover the postage, but exceeds the length restrictions of the destination post office, write a package-rejected entry to the log of the initiating post office and to the master log.
6. If there is no room in the initiating post office, write a package-rejected or letter-rejected entry to the log of the initiating post office and to the master log.

2.4 Acceptance of items

1. Accept letters if they are to be delivered to an existing post office. Write a letter-accepted entry to the initiating post office log.
2. If a package has sufficient funds to cover the postage and it meets the length requirements of both the initiating post office and of the destination post office, accept the package. Write a package-accepted entry to the initiating post office log.

3. If a package has sufficient funds to cover the postage and the persuasion amount combined, accept the package even if it is too big. Write a package-accepted entry to the initiating post office log. Write a bribery-detected entry to the master log.

2.5 Items in transit

1. At the end of each day remove all accepted items from their initiating post offices, such items are now in transit. For each item write a transit-sent entry to the initiating post office log.
2. At the beginning of each day, check if any in transit items have arrived at the receiving post office. If so, write a transit-arrived entry to the destination post office log.
3. If an in-transit package arrives at a destination post office and violates the length requirements of the destination post office, destroy it. Write a package-destroyed entry to the destination post office log and to the master log.
4. Destroy any items which arrive at a full destination post office. Write a package-destroyed or letter-destroyed entry to the destination post office log and to the master log.

2.6 Delivery errors

1. If a package has been waiting for pickup for more than 14 days, destroy it. Write a package-destroyed entry to the receiving post office log and to the master log.
2. If a letter has been waiting for pickup for more than 14 days and it does not have a return address, destroy it. Write a letter-destroyed entry to the receiving post office log and to the master log.
3. If a person on the internationally wanted criminals list attempts to pick up any item, regardless of (non)existence of such an item, write a criminal-apprehended entry to the front page log.
4. If a letter has been waiting for pickup for more than 14 days and it has a return address, return it to sender. The destination post office becomes the new initiating office, the original initiating post office becomes the new destination post office. The original sender becomes the new recipient. Avoid endless loops by making sure that the letter does no longer have a return address. From this point on, treat this letter as a newly mailed letter and follow all the regular procedures for letters.
5. Disallow attempts to pick up a nonexistent item or an item in transit. Don't log such attempts.
6. Disallow attempts to pick up an item twice. Don't log such attempts.

2.7 Successful delivery

1. During a pickup attempt the recipient picks up all items with their name at the post office of pickup. Write a item-complete entry to the destination post office log for each such item.

3 Log entry types

3.1 End-of-day

The end of day log entry consists of:

- The line ”- - DAY N OVER - -”, where N is the number of the day that has just ended.

3.2 New-letter

The new letter log entry consists of:

- The line ”- New letter -”.
- The line ”Source: X”, where X is the name of the initiating post office.
- The line ”Destination: Y”, where Y is the name of the destination post office.

3.3 New-package

The new package log entry consists of:

- The line ”- New package -”
- The line ”Source: X”, where X is the name of the initiating post office.
- The line ”Destination: Y”, where Y is the name of the destination post office.

3.4 Rejected-package

The package rejected log entry consists of:

- The line ”- Rejected package -”
- The line ”Source: X”, where X is the name of the post office which rejected the package.

3.5 Rejected-letter

The letter rejected log entry consists of:

- The line ”- Rejected letter -”
- The line ”Source: X”, where X is the name of the post office which rejected the letter.

3.6 Letter-accepted

The letter accepted log entry consists of:

- The line ”- Accepted letter -”
- The line ”Destination: X”, where X is the name of the destination post office.

3.7 Package-accepted

The package accepted log entry consists of:

- The line ”- Accepted package -”
- The line ”Destination: X”, where X is the name of the destination post office.

3.8 Package-destroyed

The package destroyed log entry consists of:

- The line ”- Incinerated package -”
- The line ”Destroyed at: X”, where X is the name of the post office which destroyed the package.

3.9 Letter-destroyed

The letter destroyed log entry consists of:

- The line ”- Incinerated letter -”
- The line ”Destroyed at: X”, where X is the name of the post office which destroyed the letter.

3.10 Bribery-detected

The bribery detected log entry consists of:

- The line ”- Something funny going on... -”
- The line ”Where did that extra money at X come from?”, where X is the name of the post office which was persuaded into accepting an oversize package.

3.11 Item-complete

The item picked up log entry consists of:

- The line ”- Delivery process complete -”
- The line ”Delivery took N days.”, where N is the count of days since the item first entered the system.

3.12 Transit-sent

The item sent log entry consists of:

- The line ”- Standard transit departure -”

3.13 Transit-arrived

The item arrived log entry consists of:

- The line ”- Standard transit arrival -”

3.14 Criminal-apprehended

Anything works. Be creative. This is the front page news. Have fun with it.

4 Input file formats

There are three types of input files that you will be dealing with.

4.1 Post office file

The name of this file is offices.txt. This file contains the information about the post offices in the system. You should read this file and use the contents to set up your program for the simulation.

The first data point in the file will be the integer count of post offices in the system.

Following the post office count integer and for each post office the following data will be provided, in order:

- A string naming the post office.
- An integer transit time for items leaving this post office.
- An integer postage required for sending packages out of this post office.
- An integer capacity of this post office.
- An integer persuasion amount of this post office.
- An integer maximum package length for this post office.

Example: Berlin 4 10 1000 10000 300 – A post office called Berlin which sends out mail that arrives in four days of transit, where packages cost 1000 units of currency, able to store 1000 items, nearly impossible to convince into shipping oversize mail and rejecting packages over 300 units of length.

4.2 Wanted criminals file

The name of this file is wanted.txt. This file contains the names of criminals which may be using the postal system for their nefarious plans to take over the world. You should read this file at program startup and thwart their evil plans by rejecting their packages or arresting them whenever they happen to be foolish enough to attempt a package pickup in person.

The first data point in the file will be the integer count of criminals at large.

Following the criminal count integer and for each criminal the following data item will be provided:

- A string naming the criminal.

4.3 Command file

The name of this file is commands.txt. This file contains instructions for your program to execute.

This file will begin with the integer count of commands within. Following will be the commands.

A command can be any of the following.

- DAY – Which signifies the end of the day.
- PICKUP – A pickup attempt. This command will be followed by the following:
  - A string naming the post office of pickup.
  - A string naming the person attempting a pickup.

  Example: PICKUP New_York Timothy – A person named Timothy is picking up mail in New_York.
  Example: PICKUP Vancouver Karol – A person named Karol is picking up mail in Vancouver.

- LETTER – A new letter being brought to an initiating office. This command will be followed by the following:
  - A string naming the post office to which the letter was brought.
  - A string naming the person who is the letter's recipient.
  - A string naming the destination post office.
  - A string naming the person who can pick up the letter if it gets returned to sender. NONE will mean that there was no return address.
    Example: LETTER Boston Khan Ulan_Baator NONE – An unknown sender in Boston is trying to send a letter to a person named Khan, to be picked up in the post office called Ulan_Baator.
    Example: LETTER Moscow Sam Washington Bill – A letter to be mailed in Moscow, with the recipient being Sam in Washington. If the letter gets returned to sender then it will be available for pickup by Bill at the post office in Moscow.
- PACKAGE – A new package being brought to an initiating office. This command will be followed by the following:
  - A string naming the post office to which the package was brought.
  - A string naming the person who is the package's recipient.
  - A string naming the destination post office.
  - An integer specifying the money accompanying this package, to be spent both on postage and on persuasion.
  - An integer specifying the length of the package.

  Example: PACKAGE Paris Pierre Toulouse 200 99999 – A package to be sent in Paris, with the destination being Toulouse and the recipient being Pierre. The package is 99999 units long and has 200 units of currency accompanying it.

5 Notes

I will put various clarifications here.

1. At program startup the current day is day 1.
2. Letters don't require postage, only packages do.
3. Packages don't have a return address.
4. Transit times are dictated solely by the initiating post office and not by the distance or any properties of the destination office.
5. The transit delay is in terms of full days. For example: A package accepted on day 3 by a post office with transit time of 5 days should enter the receiving post office at the beginning of day 9. A package accepted on day 2 with transit time of 0 should enter the receiving post office at the start of day 3.
6. For the calculation of the time in the postal system count rounding up. The packages from the above example have been in the system 7 and 2 days, respectively. Don't reset the timer if the piece of mail is returned to sender.
7. A returned to sender letter should generate the new-letter log entry and other entries appropriate for a new piece of mail. If in doubt, consult the sample inputs and outputs for handling returned mail.

8. The transit-sent log messages should precede the end-of-day log message, which in turn precedes the transit-arrived log messages.
9. Rejected packages are removed from the system and aren't sent to their destination.
10. Both letters and packages count against the storage space in a post office. Both count for one unit for storage. Use package length only for determining if it can be shipped/received by a post office.
11. To compare Strings you can use <string1>.equals(<string2>). The result will be a boolean value.