**ECMM419 Software Development for Business**
**Continuous Assessment 3**

Date set: Friday 18th December 2015
Hand-in date: **12:00 Friday 15th January, 2016**

This continuous assessment (CA) comprises 60% of the overall module assessment.

This is an **individual** exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

Note that both paper (BART) and electronic submissions are required.

---

This CA tests your knowledge of the programming in Python that we have covered in the recent lectures, particularly the writing and testing of slightly more complex programs. Additionally, you are required to produce a software design document that describes your system.

Make sure that you lay your code out so that it is readable and you comment the code appropriately.

## The Project

The London underground system consists of over 250 miles of track on which more than 1 billion customer journeys are made every year. To make this complex system easier to use, Transport for London have recently decided to install a new program on their ticket machines that:

- *requires* a customer to state the station where their journey will begin and end;
- *plans* the best route between these stations, and calculates the fare;
- *displays* the tube network with the optimum route highlighted (e.g., using a thicker line or a different colour);
- *accepts* payment in 1p, 2p, 5p, 10p, 20p, 50p, £1 and £2 coins;
- *issues* a ticket with a route plan printed on it, along with change in the smallest number of 1p, 2p, 5p, 10p, 20p, 50p, £1 and £2 coins.

So, for example, a customer wishing to travel between Paddington and Oxford Circus, tendering two £1 coins, might be issued with a ticket with the route plan

$$\text{Paddington} \xrightarrow{Circle} \text{Baker Street}$$
$$\text{Baker Street} \xrightarrow{Bakerloo} \text{Oxford Circus}$$

printed on it, along with a 50p coin in change.

The aim of the project is to develop software for a prototype program for the ticket machines.

# What Information is Provided?

Three files are provided: stations.xml, line.xml and zones.xml. These files contain up-to-date route information.

## 1 Stations

The stations database is a large xml file, with records structured as follows:

$$
\begin{aligned}
&< Station\ id = \ ''Aldgate'' > \\
&\quad < Latitude > 51.51394 < /Latitude > \\
&\quad < Longitude > -0.07537 < /Longitude > \\
&\quad < Line > [Metropolitan] < /Line > \\
&\quad < Zone > 1 < /Zone > \\
&< /Station >
\end{aligned}
$$

The example above gives the latitude (51.51394) and longitude (-0.07537) of Aldgate station, in addition to specifying that it is on the Metropolitan line and in zone 1.

Important points to note are:

- the database may contain any number of stations, including zero;
- if a station is on more than one line, the lines will be separated by ‘,’. For example the line field within Waterloo would be represented by [Bakerloo,Northern, Waterloo & City];
- a station must have a name, and be connected by at least one line.

## 2 Lines

The lines database records the order in which the stations occur on each line.

$$
\begin{aligned}
&< Line\ id = \ \text{“}Waterloo\&City'' > \\
&\quad < Station1 > \text{“}Waterloo\text{”} < /Station1 > \\
&\quad < Station2 > \text{“}Bank\text{”} < /Station2 > \\
&< /Line >
\end{aligned}
$$

This record states that Waterloo station is next to Bank station on the Waterloo & City line.

Important points to note are:

- the database may contain any number of lines, including zero;
- a line must have at least two stations on it.

### 3 Zones

The zones database records the cost of travelling within and between zones. Here is an example record:

$$
\begin{array}{l}
< Zone\ id = \ ''Zone1'' > \\
\quad < Zone1 > 1.20 < /Zone1 > \\
\quad < Zone2 > 1.80 < /Zone2 > \\
\quad < Zone3 > 2.00 < /Zone3 > \\
\quad < Zone4 > 2.40 < /Zone4 > \\
\quad < Zone5 > 2.80 < /Zone5 > \\
\quad < Zone6 > 3.20 < /Zone6 > \\
\quad < Zone7 > 3.50 < /Zone7 > \\
\quad < Zone8 > 4.00 < /Zone8 > \\
\quad < Zone9 > 4.80 < /Zone9 > \\
< /Zone >
\end{array}
$$

This example shows that it costs £1.20 to travel within zone 1, £2.00 to travel between zone 1 and zone 3 and £4.80 to travel between zone 1 and zone 9.

Important points to note are:

- the database may contain any number of zones, including zero;
- a zone must have a name and connection costs to eight other zones.

## System Requirements

The aim of the project is to develop software for the prototype ticket machines. The software should perform the following tasks.

1. Read the station and zone databases into memory from the file store. These database files are not supposed to contain errors, and so any errors should cause the program to halt immediately with an appropriate message.

2. Prompt the customer for the names of the stations where the journey will begin and end, and find the best route between them.
   - *The precise details of "best route" are for you to determine, and to clearly specify.*

3. Prompt the customer for the correct fare, and read the amount tendered from the keyboard.
   - *You should assume payment in cash without concession. The complexity introduced by weekly off-peak travel cards and so on is considerable!*

4. Print a "ticket" on the screen with a route plan on it, and the amount of change due to the customer.
    - *The description of the route is up to you, but it must be clear which lines are to be taken, and where the changes are to be made.*
5. Using Turtle Graphics display a map of the tube network with the chosen route highlighted (e.g., the route could be displayed with a thicker line or a different colour).

All user input will be via the terminal.

## Your Task

Your task is to employ the software engineering and programming skills you have acquired during the module to design and implement a prototype system for Transport for London. You may develop using either a functional design or object orientation.

As well as a working prototype that fulfils the system requirements described above you must present a detailed design specification, including:

- Diagrammatic overviews of your system (if you follow a functional design you should include at least an entity relationship diagram and appropriate data flow diagrams; if using an object oriented design you should include a class diagram, a use case diagram and appropriate sequence diagrams).
- A function catalogue (or description of the methods each class will use, if developing an object oriented system).
- Pseudo code for computing the best route between two stations.
- A comprehensive test plan describing how you would test the various aspects of your system.

## Submitting your work

The CA requires both paper and electronic submissions.

**Paper** You should submit electronic and paper copies of your design document, as well as the code and any output to the Harrison Student Services Office by the deadline of **12:00 Friday 15th January, 2016**. Markers will not be able to give feedback if you do not submit hardcopies of your code and marks will be deducted if you fail to do so.

Paper submissions should have the BART cover sheet securely attached to the front and should be anonymous (that is, the marker should not be able to tell who you are from the submission).

Where you are asked for paper copies of the output of your code, please copy and paste the output from the terminal rather than taking a screenshot, because the screenshot is often illegible after printing. To cut and paste from a Windows Command window, highlight the text you want to paste, right click in the Command window, click on "Select all", press Enter, and paste into your document (control-V or from the menu).

**Electronic** You should submit the files containing the design document and code via the electronic submission system at `http://empslocal.ex.ac.uk/submit/`. To do this, use `zip` or `rar` or `tar` to compress these into a single file, and upload this file using the submit system. You must do this by the deadline.

You will be sent an email by the submit system asking you to confirm your submission by following a link. Your submission is not confirmed until you do this. It is best to do it straightaway, but there is a few hours leeway after the deadline has passed. It is possible to unsubmit and resubmit electronic coursework — follow the instructions on the submission website.

## Marking criteria

Work will be marked against the following criteria.

- **Does you design document fully describe the system?**
  - Do the diagrams you have provided give a suitable overview of the system?
  - Is your function catalogue complete?
  - Does your test plan test all aspects of the system?
- **Does your algorithm correctly solve the problem? Is it succinctly described with pseudo code?**
- **Does the code correctly implement the algorithm and specification?**
- **Is the code of good quality?**

There is a 10% penalty for not submitting hardcopies of your programs.