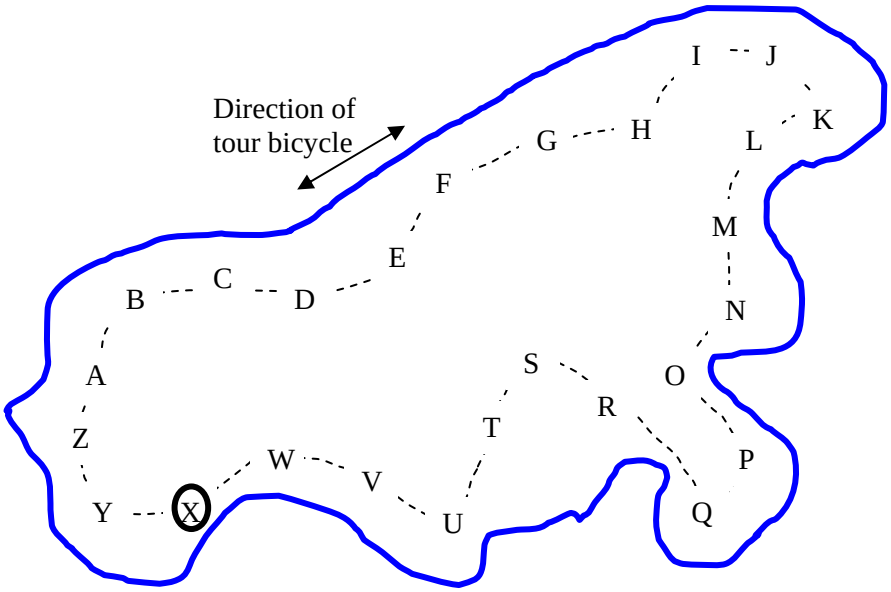
ENGG1002D Computer Programming and Applications - Assignment 2
First Semester, 2012-2013

The Problem

With Island Bicycle Tour, tourist can sit and relax on a tour bicycle while the tour guide drives along the scenic path of Lantau Island. A tour map of Lantau Island with attraction points is shown below:



There are 26 attraction points in the island. For simplicity, we are going to use the upper case alphabets to name these attraction points (i.e. A, B, C, …, Y, Z). These attraction points line up in the same order as the alphabets, therefore a complete island tour starting at A >> B >> C >> . . . >> Y >> Z >> A will let a tourist enjoy everything on the island.

Operation of bicycle tour

All bicycles (assuming bicycles are big enough to carry all tourists) are located at attraction point **X**. Each day, tour guides will get their tour lists here at point X, and start their tour drive at this point. At the end of the day, they will return their bicycles to X for their pay. For example, John, the tour guide, has a tour list like this:

Tour list for John	What does it mean?	What will the tour guide (John) do?
3AD	3 tourists want to go from A to D.	Drive his empty bicycle from X to A, pick up 3 tourists, and drive them to D, stopping at B, C and D. Drop off the tourists at D.
4EG	4 tourists want to go from E to G.	Drive his empty bicycle from D to E, pick up 4 tourists, and drive them to G, stopping at F and G. Drop off the tourists at G.
6MJ	6 tourists want to go from M to J.	Drive his empty bicycle from G to M, pick up 6 tourists, and drive them to J, stopping at L, K and J. Drop off the tourists at J.
		Drive his empty bicycle from J to X for his pay.

We are going to assume the distance between any 2 adjacent attraction points is the same for all adjacent pairs of attraction points. When the tour guide drives from one attraction point to another attraction point, he / she always takes the shortest path, so driving from X to A will be X >> Y >> Z >> A, and driving from M to J will be M >> L >> K >> J.

To calculate the total time spent by a tour guide for the tour, the following formulae are used:

> Time taken for a drive = Energy * (number of people) * number of stops
> Time taken for a tour = time taken for a drive + 500 * number of tourists

For example, to calculate the total time taken for John to finish his job, we need to consider every drive he has taken, which includes the time taken for John to drive his empty bicycle from one point to another point. As an example, if the energy of John is 200, the time taken for each tour and the accumulated time can be calculated as below:

Tour	**Number of stops**	**Time taken**	**Accumulated time**
From X to A	3	200 * 1 * 3 = 600	600
3AD	3	200 * 4 * 3 + 500 * 3 = 3900	4500
From D to E	1	200 * 1 * 1 = 200	4700
4EG	2	200 * 5 * 2 + 500 * 4 = 4000	8700
From G to M	6	200 * 1 * 6 = 1200	9900
6MJ	3	200 * 7 * 3 +500 * 6 = 7200	17100
From J to X	12	200 * 1 * 12 = 2400	19500

Your task

The company Island Bicycle Tour wants you to write a program to keep track of the accumulated time and calculate a few statistics so that the boss knows how much to pay the tour guides. For simplicity, you only need to consider John (the only tour guide) and his bicycle (only 1 bicycle). Your program should observe the following input format:

Line 1: energy of the tour guide
Line 2 onwards: a list of tours terminated by the tour entry: **`0XX`**

This is zero

a tour has this format: P P_1 P_2
P = number of tourists
P_1 = attraction point where the tour starts
P_2 = attraction point where the tour ends

Your task is to compute the payment index for the tour guide. Calculation is shown below:

> Payment index = T / (1 + P_{total} + S)
> Where T = total time taken for the tour guide to finish his job
> P_{total} = total number of tourists served
> S = total number of attraction points visited by the tourists served

In our example, T = 19500, P_{total} = 13, S = 11, therefore payment index for John is 780.

When an input tour entry contains invalid data, display a message "`error, skipping this tour`", skip that tour and continue with the next tour. The following input is considered invalid:

- energy is zero or negative
- number of tourists is zero or negative
- attraction point is not an uppercase letter
- starting and ending points are the same

But you may assume the following:

- first line is an integer
- first field of the tour entry is an integer (can be bigger than 9)
- second and third fields of the tour entry is a character

Sample input and output

Below are some sample runs. Input is shown in **bold type**. Please take note of what to output in different scenarios. Explanation in the second column will help you understand the detail operations for each tour. Note that this is not the output of your program.

Sample **input** and output	**Explanation (for your reference only)**
200 **3AD** driving X to A accumulated time: 600 driving 3 tourist A to D accumulated time: 4500 **4EG** driving D to E accumulated time: 4700 driving 4 tourist E to G accumulated time: 8700 **6MJ** driving G to M accumulated time: 9900 driving 6 tourist M to J accumulated time: 17100 **0XX** back to base X accumulated time: 19500 Total tourists: 13 Total attractions: 11 Payment index: 780	energy=200 3 tourists want to go from A to D. John drives the empty bicycle to A. John drives 3 tourists to D. 4 tourists want to go from E to G. John drives the empty bicycle to E. John drives 4 tourists to G. 6 tourists want to go from M to J. John drives the empty bicycle to M. John drives 6 tourists to J. All tours finished. Back to base. John drives the empty bicycle to X. Final statistics.
100 **0AD** error, skipping this tour **3CB** driving X to C accumulated time: 500 driving 3 tourist C to B accumulated time: 2400 **1JJ** error, skipping this tour **0XX** back to base X accumulated time: 2800 Total tourists: 3 Total attractions: 2 Payment index: 466.667	energy=100 This tour is not valid. 3 tourists want to go from C to B. John drives the empty bicycle to C. John drives 3 tourists to B. An invalid tour again All tours finished. Back to base. John drives the empty bicycle to X. Final statistics.

Sample **input** and output	**Explanation (for your reference only)**
300 **0XX** back to base X accumulated time: 0 Total tourists: 0 Total attractions: 0 Payment index: 0	energy=300 No tour. Back to base. Final statistics.
300 **12XD** driving 12 tourist X to D accumulated time: 29400 **1DX** driving 1 tourist D to X accumulated time: 33500 **0XX** back to base X accumulated time: 33500 Total tourists: 13 Total attractions: 14 Payment index: 1196.43	energy=300 12 tourists want to go from X to D. Bicycle is already at X, John drives 12 tourists to D. 1 tourist want to go from D to X. Bicycle is already at D, John drives 1 tourist to X. All tours finished. Back to base. Final statistics.

Program logic

The program needs to keep track of the accumulated time, total number of tourists, total number of attractions and payment index. From the above samples, we noticed that the accumulated time must be updated whenever the bicycle stops at a location (with or without tourist). To start thinking, let's see how John, the tour guide proceeds with his daily routine:

```
Start his day at X
Read the first instruction in his tour list
While this instruction is not 0XXn
    If the information valid?
        Update total number of tourists and total number of attractions
        Drive his bicycle to the starting point
        Update accumulated time
        Take tourists to the end point
        Update accumulated time
    Display error message if the information is not valid.
    Read the next instruction
Now, all his jobs had completed.  Go back to X.
Update accumulated time
Calculate payment index
```

Useful library function

To compute the absolute value, we can write a user defined function, or make use of library function:

Function name	`abs`
Which library is needed	`cstdlib`
Function argument	an integer
Return data type	an integer
An example	`int x, y=-5;` `x = abs (y);` // x = 5, which is the absolute value of y

Program style

Besides program correctness, marks might be deducted if the followings are not followed:

- I/O format - your program will be marked by a program, therefore the wordings, format and order of input / output must follow exactly that of the above examples.
- Program style - your program should be easy to read and understand. Make use of proper indentation, comments and meaningful variable names to achieve this.
- Programming skills - only those skills being taught up to lecture 5 can be used. Array and **break** cannot be used.
- Correct submission - Name the cpp files as requested and include necessary information (e.g. UNo) in each program.

Marking scheme

The assignment will be marked in the following way:

- [10%] Your program should define and make use of at least 2 user defined functions to reduce duplications and unnecessary complexity of the program logic.
- [90%] Your program will be tested against a number of cases. Each case will be worth a certain number of marks

Rules and regulations

[**Plagiarism is strictly prohibited**] Zero marks will be given to both the source and copy if discovered. You must not copy or let others copy your work. It's your own responsibility to prevent others from copying your program directly or indirectly (e.g. obtain your program through another person).

The following constitutes an act of plagiarism:

- submitting the work of another student as your own, or asking someone else to do your assignment;
- direct copying;
- studying someone else's program and then rewriting it as your own;
- using a substantial part of other's work in your program;
- providing your assignment as the source for any of the above actions.

This assignment is an individual work. You can discuss with others in the design phase only. Coding must be done separately and group program is NOT accepted.
[Late policy] Late submission will not be marked.

Submission (Late submission will not be marked)

Only limited test cases will be made public for your testing. You are encouraged to create your own test cases for testing purpose. Your program will be tested against a number of cases, and each case will be worth a certain number of marks.

What to submit?

Include your 10-digit UNo as part of the filename and also as comment inside the program. Submit only the **source program** (i.e. *yourUNo*_**a2.cpp**). Please note that programs developed on other platform may not work properly under Dev-C++. Marking will be done using Dev-C++ version 4.9.9.2.

Where to submit?
Email your source programs as attachments to engg1002@hku.hk before the deadline. Copy the email to yourself as evidence in case the marker couldn't receive your work. Make sure you write your 10-digit UNo as part of the filename. Attachment without UNo will be ignored. Indicate you are submitting assignment 2, and in case you want to submit a newer version before the deadline, please indicate in the subject as well.

When to submit?
Submission deadline is **5:00pm 26 Oct (Fri)**. Late submission will not be marked.