ENGG1111 Computer Programming and Applications - Assignment 2
Second Semester, 2016-2017

Where is the most popular tourist destination?

In this assignment, we are going to collect statistics and hopefully can answer this question.

Joyful Travel Agent

This is a small local travel agent in HK. To gain competitive advantage, the marketing manager Kate came up with a campaign "Get the lowest price in town from us". Through Joyful travel, customers can buy tickets offered by any airline at a 10% discounted price. At the back end, marketing team will be collecting statistics for future marketing plans. As a technical support staff, you are going to construct a program to provide various supporting operations for this campaign.

Supporting operations

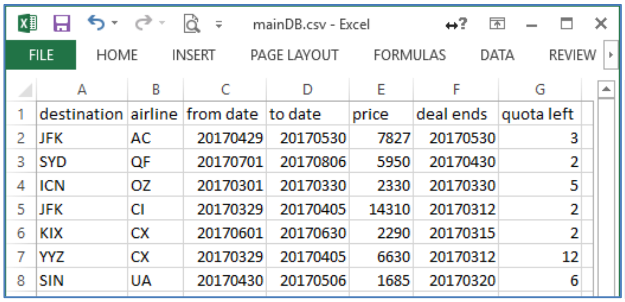
Below are the supporting operations for this campaign:

- Read the main database ("mainDB.csv"). This database contains ticketing information.
- Add / update entries.
- Remove outdated information and not useful information.
- Display ticketing information.
- Buy ticket and update quota.
- Search and refine search functions.
- Find out where is the most popular tourist destination.

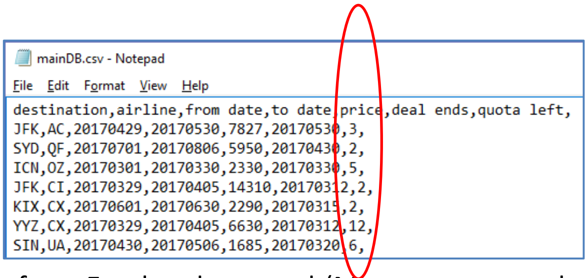
To produce a smooth daily operation, you will be working with these data files:

Input filename: `mainDB.csv`

sorry, no comma at the end of line



	A	B	C	D	E	F	G
1	destination	airline	from date	to date	price	deal ends	quota left
2	JFK	AC	20170429	20170530	7827	20170530	3
3	SYD	QF	20170701	20170806	5950	20170430	2
4	ICN	OZ	20170301	20170330	2330	20170330	5
5	JFK	CI	20170329	20170405	14310	20170312	2
6	KIX	CX	20170601	20170630	2290	20170315	2
7	YYZ	CX	20170329	20170405	6630	20170312	12
8	SIN	UA	20170430	20170506	1685	20170320	6

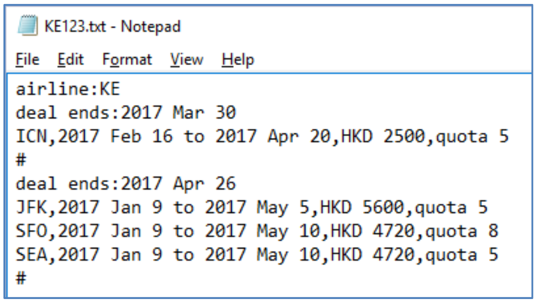


```
destination,airline,from date,to date,price,deal ends,quota left,
JFK,AC,20170429,20170530,7827,20170530,3,
SYD,QF,20170701,20170806,5950,20170430,2,
ICN,OZ,20170301,20170330,2330,20170330,5,
JFK,CI,20170329,20170405,14310,20170312,2,
KIX,CX,20170601,20170630,2290,20170315,2,
YYZ,CX,20170329,20170405,6630,20170312,12,
SIN,UA,20170430,20170506,1685,20170320,6,
```

The above diagrams show the view of **mainDB.csv** from Excel and notepad (Mac users: open the file in Numbers or TextEdit). This database contains ticketing information gathered from airlines. The first line is the header. From the second line onwards, each line corresponds to one travel promotion (assuming economy return ticket flying from HK):

1. Airport code of the travel destination
2. Airline code of the airline
3. Travel must begin on or after the **from date** (yyyymmdd).
4. Travel must be completed on or before the **to date** (yyyymmdd).
5. Price of this ticket (a further 10% discount if customer buy through Joyful Travel)
6. The last date (yyyymmdd) customer can buy this ticket to enjoy the promotion offered.
7. Quota remaining for this promotion.

Input filename: filename entered by operator



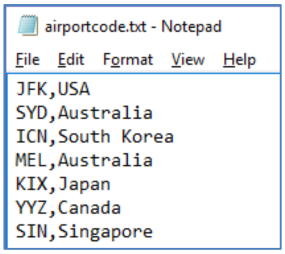
```
airline:KE
deal ends:2017 Mar 30
ICN,2017 Feb 16 to 2017 Apr 20,HKD 2500,quota 5
#
deal ends:2017 Apr 26
JFK,2017 Jan 9 to 2017 May 5,HKD 5600,quota 5
SFO,2017 Jan 9 to 2017 May 10,HKD 4720,quota 8
SEA,2017 Jan 9 to 2017 May 10,HKD 4720,quota 5
#
```

This file contains promotion information offered by an airline. The format of this file is a bit different from that of the main database. The first line specify the name of the airline. The second line specify the last date of this promotion deal, followed by a few promotion items having the same end-of-deal date. Since the promotion information comes from an airline, it is understood that all flights are operated under that airline, therefore name of airline is omitted in the promotion details. # marks the end of promotion items having the same end-of-deal date.

These promotion items will be added to the end of the main database if they are new. If the promotion item already exists in the database, simply add the quota to the existing record. Do not sort. Just append new records in the same order as they appear in the input file.

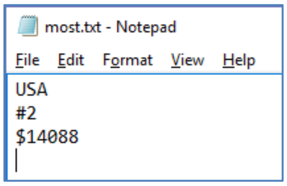
Input filename: `airportcode.txt`

This file maps airport code to its country. Each line has a 3 character airport code and a country name separated by a comma. There is a maximum of 200 entries in this file. In general, it is possible for a country to have more than one airport.



```
JFK,USA
SYD,Australia
ICN,South Korea
MEL,Australia
KIX,Japan
YYZ,Canada
SIN,Singapore
```

Output filename: `most.txt`



```
USA
#2
$14088
```

This file answers our question: where is the most popular tourist destination? Each time when a ticket is purchased through Joyful Travel, the system will accumulate the number of tickets sold, total revenue, and destination country. When asked by the operator (command **most**), the system will be able to write answer to an output file. In the case there is more than one country having the highest number of tickets sold, output any one.

Format:
Line #1: country name
Line #2: total number of tickets sold (put a # before the number)
Line #3: total price of the tickets sold (put a $ before the revenue)
(The vertical stroke shown in the above diagram represents the position of cursor after writing line #3.

This system models the daily ticketing of Joyful Travel, therefore tickets bought by customers cannot be made available to other customers. In other words, when the operator turns on the computer the next day and run the system, the system should behave as if it has never been turned off. We need an intermediate file to hold useful information so that the system can continue as usual on the next day. Besides, the main database is a read only file shared among staffs.

What kind of information to record? You are going to design this intermediate file.

Design an intermediate data file to store useful ticketing information

Design of this file will affect the program logic. A sample design is given as a reference in the testing package. Note, the sample design only worth 5/20 marks. Please take into considerations the pros and cons in your design. If needed, you may create up to 2 intermediate data files.

Your task

Develop a program which provides the following operations to support the campaign.

Command (all commands are entered in a single line)	**What to do**
`add promo` *filename*	Add promotion information to the main database from an input file (filename entered by the operator). Format of the input data file is described above. If an item is new, add to the end of the database. If an item exists in the database, add quota to existing record.
`all`	Display all promotion information. This command will not be marked. Use it for debugging.
`today` *yyyymmdd*	Remove all promotion information having the end-of-deal date earlier than this input date, where *yyyymmdd* represents a valid date.
`clear`	Clear all marked records. Do this command before a new search.
`search dest` *xxx*	Search and display all promotion information having the same travel destination as *xxx*. Selected records are numbered starting from 1, and displayed according to the order in the main database.
`search date` *yyyymmdd*	Search and display all promotion information having the date *yyyymmdd* lies within the travel period. Selected records are numbered starting from 1, and displayed according to the order in the main database.
`prefer airline` *xxx*	Refine the search by selecting all records operated by airline *xxx*. Display the refined results. Selected records are numbered starting from 1, and displayed according to the order in the main database.
`no airline` *xxx*	Refine the search by not selecting all records operated by airline *xxx*. Display the refined results. Selected records are numbered starting from 1, and displayed according to the order in the main database.
`buy #`*k n*	Buy *n* tickets of item *k*, where k is the *k*-th item displayed in the search just above this operation. In case there is not enough ticket, do not perform this operation. search must be performed before **buy #***k n*.

Command (all commands are entered in a single line)	**What to do**
buy *dest-airline-fromDate-toDate-price n*	Find the promotion item having the requested destination, airline, travel period and price, then buy *n* tickets. You can assume this selection will give zero or one match. In case there is not enough ticket, do not perform this operation. e.g. buy KIX-CX-20170601-20170630-2290 1 buy 1 ticket operated by CX to KIX, travel period: 20170601 to 20170630, $2290.
most	For all available countries in **airportcode.txt**, calculate the total number of tickets sold and total revenue, then identify the destination country with the largest number of tickets sold. Write to **most.txt**
end	End the program.

Program testing

Sample executable and data files are available in Moodle for testing purpose. Please download from Moodle according to your Operating System. Do not attempt to reverse engineer the executable file.

Working with data files

If you are using Windows OS,

- Please use ONLY the data file for Windows provided in Moodle.
- Put all the data files in the same folder as your cpp file.

If you are using Mac OS,

- Please use ONLY the data file for Mac provided in Moodle.
- Put all the data files in your home folder.

The following shows two sample runs using data in this document. Please use the executable file in Moodle to generate more samples.

#1 Sample (cin **bolded**)	Output file - most.txt

```
buy YYZ-CX-20170329-20170405-6630 5
5 ticket(s) purchased
buy SYD-QF-20170701-20170806-5950 10
not enough tickets
search dest JFK
  <1> JFK-AC-20170429-20170530 HKD 7827, promotion expires 20170530, 3 quota
  <2> JFK-CI-20170329-20170405 HKD 14310, promotion expires 20170312, 2 quota
buy #1 3
3 ticket(s) purchased
most
end
```

```
Canada
#5
$29835
```

Note: all cout will not be marked. So, don't worry about the cout formatting. Use cout for debugging purpose.

An intermediate file was created after running sample #1. Now, using intermediate file, we run the program again. Input and output are shown in sample #2 in the next page.

#2 Sample (cin **bolded**) continues from above sample

```
all
~~[0] JFK-AC-20170429-20170530 HKD 7827, promotion expires 20170530, 0 quota~~
[0] SYD-QF-20170701-20170806 HKD 5950, promotion expires 20170430, 2 quota
[1] ICN-OZ-20170301-20170330 HKD 2330, promotion expires 20170330, 5 quota
[2] JFK-CI-20170329-20170405 HKD 14310, promotion expires 20170312, 2 quota
[3] KIX-CX-20170601-20170630 HKD 2290, promotion expires 20170315, 2 quota
[4] YYZ-CX-20170329-20170405 HKD 6630, promotion expires 20170312, 7 quota
[5] SIN-UA-20170430-20170506 HKD 1685, promotion expires 20170320, 6 quota
total 6 records
add promo KE123.txt
all
[0] SYD-QF-20170701-20170806 HKD 5950, promotion expires 20170430, 2 quota
[1] ICN-OZ-20170301-20170330 HKD 2330, promotion expires 20170330, 5 quota
[2] JFK-CI-20170329-20170405 HKD 14310, promotion expires 20170312, 2 quota
[3] KIX-CX-20170601-20170630 HKD 2290, promotion expires 20170315, 2 quota
[4] YYZ-CX-20170329-20170405 HKD 6630, promotion expires 20170312, 7 quota
[5] SIN-UA-20170430-20170506 HKD 1685, promotion expires 20170320, 6 quota
[6] ICN-KE-20170216-20170420 HKD 2500, promotion expires 20170330, 5 quota
[7] JFK-KE-20170109-20170505 HKD 5600, promotion expires 20170426, 5 quota
[8] SFO-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 8 quota
[9] SEA-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 5 quota
total 10 records
search date 20170315
  <1> ICN-OZ-20170301-20170330 HKD 2330, promotion expires 20170330, 5 quota
  <2> ICN-KE-20170216-20170420 HKD 2500, promotion expires 20170330, 5 quota
  <3> JFK-KE-20170109-20170505 HKD 5600, promotion expires 20170426, 5 quota
  <4> SFO-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 8 quota
  <5> SEA-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 5 quota
no airline OZ
  <1> ICN-KE-20170216-20170420 HKD 2500, promotion expires 20170330, 5 quota
  <2> JFK-KE-20170109-20170505 HKD 5600, promotion expires 20170426, 5 quota
  <3> SFO-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 8 quota
  <4> SEA-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 5 quota
buy #3 7
7 ticket(s) purchased
most
all
[0] SYD-QF-20170701-20170806 HKD 5950, promotion expires 20170430, 2 quota
[1] ICN-OZ-20170301-20170330 HKD 2330, promotion expires 20170330, 5 quota
[2] JFK-CI-20170329-20170405 HKD 14310, promotion expires 20170312, 2 quota
[3] KIX-CX-20170601-20170630 HKD 2290, promotion expires 20170315, 2 quota
[4] YYZ-CX-20170329-20170405 HKD 6630, promotion expires 20170312, 7 quota
[5] SIN-UA-20170430-20170506 HKD 1685, promotion expires 20170320, 6 quota
[6] ICN-KE-20170216-20170420 HKD 2500, promotion expires 20170330, 5 quota
[7] JFK-KE-20170109-20170505 HKD 5600, promotion expires 20170426, 5 quota
[8] SFO-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 1 quota
[9] SEA-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 5 quota
total 10 records
today 20170401
all
[0] SYD-QF-20170701-20170806 HKD 5950, promotion expires 20170430, 2 quota
[1] JFK-KE-20170109-20170505 HKD 5600, promotion expires 20170426, 5 quota
[2] SFO-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 1 quota
[3] SEA-KE-20170109-20170510 HKD 4720, promotion expires 20170426, 5 quota
total 4 records
end
```

Output file - most.txt

```
USA
#10
$50868
```

Assumptions you can / cannot made in the program

You can base on the following assumptions when writing the program:

- All input data files exist. They are non-empty and contain at least one line. The input data file used in the command **add promo** can be any name entered by the operator.
- There will be no empty line at the end of all input files.
- Maximum 200 lines in airportcode.txt.
- Maximum 500 promotion items after executing each command. That is, total number of active records in the main database will not exceed 500 after every command.
- In the program, ~~it is ok to physically update mainDB.csv or~~ read everything into the computer memory and work on it. (Note: it's easier to mess up the data if we physically update mainDB.csv in the program. So, we consider mainDB.csv as a read only file.)

- Data files given are samples only. The data files used for marking have the same format but different contents.
- All user input will be in correct format.
- Date will always be valid and not restricted to 2016, 2017.
- In each test run, the last command will always be "end".
- In a sequence of test runs, date entered by user will always go in a forward direction. For example, command ***today** 20170215* will be entered before command ***today** 20170312*.
- Appropriate size of array must be created. Do not create oversized array.
- Do not assume any default initialization of variables and array.
- No need to generate error message. If your program considers the input is wrong, just ignore that input.
- Screen output will not be marked. Make good use of cout for debugging.
- Output file (most.txt) will be marked by computer, formatting must be observed strictly.
- Apart from the required input and output files, do not create more than 2 intermediate data files.

Rules and regulations

[I/O formats] Your program will be marked by machines. Therefore, the given input (from keyboard, from file) and output (to file) formats should be followed strictly. All cout will not be marked by computer. Each test case will be marked as "correct" or "incorrect", no partial marks will be given to a test case.

[Programming skills] You must demonstrate the use of struct, arrays, functions, strings, and file in your program. Programming skills outside the scope (lecture 1-8) cannot be used. Improper logic, such as using "break" to force a loop to terminate, will cause mark deduction. If you are not sure, please ask.

[Plagiarism] Plagiarism is strictly prohibited! Maximum penalty: zero for the coursework component in this course to both the source and copy if discovered. You must not copy or let others copy your work. It's your own responsibility to protect your files and 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 another student's work as your own, or outsourcing;
- direct copying (full or partial);
- studying someone else's program and then rewriting it as your own;
- group work (i.e. working side by side, and discuss the program line by line);
- providing your assignment as the source for any of the above actions.

Submission (Late submission will not be marked)

Only limited test cases are made public for testing. You are encouraged to create other 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.

Submission of your program (*yourUNo*_a2.cpp**)** by **6:00pm 24 Apr (Mon)** via **Moodle**
Put your name and your 10-digit UNo as comment in the first line of your cpp program. Submit a single **.cpp file** (i.e. *yourUNo*_a2.cpp). Unless Moodle is proved to be out of service within 30 minutes before the deadline, the deadline will not be extended under all circumstances.

Submitting a wrong file is not an excuse for late submission. Make sure you have double checked before submission.

Submission of your paper report during lecture on **27 Apr (Thu)**
Prepare the followings (not more than 2 pages of one A4 paper):

- Your name, university no, email.
- Screen captures showing: (1) all function declarations (the first line of every function), (2) user defined structs, (3) major arrays declared in main and global.
- Screen capture of your user defined file. Explain your design by analysing the pros and cons of the file design, and how this design affects the program effectiveness.
- [Bonus] Share 1 smart feature of your program **OR** write a smart add-on command (including program code) to get the bonus.

Marking scheme [total mark: 100]

- 60 marks for program testing
    - 30 marks (do this first): today, buy, most
    - 20 marks: search, prefer, no, clear
    - 10 marks: add
- 20 marks for design and use of the intermediate data file. More marks will be given for a small and effective design.
- 10 marks for design and use of functions. More marks will be given for:
    - Passing array to function instead of using global array
    - Meaningful functions to break down the logic
- 10 marks for the report.
- 10 marks for the bonus (must complete all operation commands).
- Deduction (max 20 marks) for incorrect input/output format, wrong filename.
- Deduction (max 50 marks) for the use of break or skills outside our scope.

Programming tips

- Work on tutorial 7 before attempting this assignment.
- Make sure the last line is not empty for input data files. After editing .csv in Excel, an empty will be created at the end, please remove it in Notepad before using it on the program.
- Use `cout` to confirm all data had been correctly read from the data files before working.
- stringstream is a bit complicated when reuse. Create the stringstream inside the loop, then we have a new stringstream to work on in every iteration.
- When using `.find()`, always compare with -1 directly (e.g. `if (text.find(target) != -1`). This is because, when the `target` does not exist in the `text`, the return value is a keyword (`string::npos`), which is logically equals to -1 (gives true when `== -1`), but it does not physically stores the value -1.
- intermediate file
    - think carefully, as this will affect the program logic.
    - The program will perform read and write to this file. Close the file before open again.
    - Usually, we write
      `fout << "some outputs…." << endl;`
      for file, reading will be easier if we write
      `fout << endl << "some outputs….";`