# MCS 260 Project Two : Storing Numbers with a Time Stamp due Wednesday 5 July at 10am

The goal of this project is to write a Python program to store numbers (as floating-point numbers) with a time stamp. At the start, the program asks to add or retrieve data.

1. To add a number, the user is prompted for a number. The program then computes the time stamp (with `ctime` of the `time` module), and stores the string representation of tuple, where the first item in the tuple is the time stamp, the second item is the number. The key of each item is the string representation of an integer, which equals the number of items when the item is added.
2. To retrieve a number, the user is shown a prompt with the count of the items in the dbm file. The program then shows the number, followed by the string `'was stored at'`, followed by the time stamp.

As name of the dbm file, you can use `datednumbers`. You may assume the user of your program does not make type errors and does not ask to retrieve items that are not there.

If the program is saved in the file `datednumbers.py`, then sessions in a terminal window (using the `$` as symbol for the prompt in the terminal) could go as follows:

```
$ python datednumbers.py
Welcome to our dated numbers!
-> type a to add, r to retrieve : a
-> give a number : 98.2
stored ('Sat Jun 24 12:47:52 2023', 98.2)
$ python datednumbers.py
Welcome to our dated numbers!
-> type a to add, r to retrieve : r
-> enter a natural number < 1 : 0
('Sat Jun 24 12:47:52 2023', 98.2)
$ python datednumbers.py
Welcome to our dated numbers!
-> type a to add, r to retrieve : r
-> enter a natural number < 1 : 0
98.2 was stored on Sat Jun 24 12:47:52 2023
$ python datednumbers.py
Welcome to our dated numbers!
-> type a to add, r to retrieve : a
-> give a number : 88
stored ('Sat Jun 24 12:49:31 2023', 88.0)
$ python datednumbers.py
Welcome to our dated numbers!
-> type a to add, r to retrieve : r
-> enter a natural number < 2 : 1
88.0 was stored on Sat Jun 24 12:49:31 2023
$
```

No special formatting of the numbers is needed.

Some important points:

1. Do not submit programs that do not run. It is much better to submit an incomplete program that runs than a program that does not run.
2. You may assume that the users of your program are on their best behavior and enter numbers on input.
3. The layout of the dialogue with the user and the formatting of the results must be exactly as in the example above.
4. The first line of your Python program must be

   ```
   # MCS 260 Project Two by <Author>
   ```

   where you replace the `<Author>` by your name.
5. Add documentation to clarify your choice of variables and to indicate the steps of the program.
6. The computer project must be solved *individually*. Collaborations are not allowed.
7. Upload your solution as a file with the `.py` extension into gradescope before 10am on Wednesday 5 July.
8. The late deadline is 5pm on Wednesday 5 July, but solutions that are submitted late are subject to a penalty of 10 points off.

If you have questions or difficulties with the project, please ask for help during office hours.