Question # 1

We want to create a simple logging facility where N process will log messages to a log file - logfile.txt. All processes will log messages for an indefinite period of time. The processes will come in at random times, open the log file, log a message and wait for a random time before coming in again for logging. The number of processes that can write to the log file will be a received via command line. The log file is a shared resource. You have to ensure that a message from one process does not get interleaved with the message from another.

Message format:

PID = <PID>; Current local time and date: <Current time and date>

Sample log file

```
PID = 1234; Current local time and date: Fri Dec  7 12:52:46 2018
PID = 1235; Current local time and date: Fri Dec  7 12:52:47 2018
PID = 1236; Current local time and date: Fri Dec  7 12:52:48 2018
PID = 1237; Current local time and date: Fri Dec  7 12:52:48 2018
PID = 1236; Current local time and date: Fri Dec  7 12:52:50 2018
PID = 1234; Current local time and date: Fri Dec  7 12:52:50 2018
PID = 1235; Current local time and date: Fri Dec  7 12:52:52 2018
```

Note: The C/C++ code for getting current local date and time has been provided

Question # 2

In this question you have to write a menu driven program that creates threads to modify and display records from a memory- mapped file.

A file carinfo.bin contains the following attributes related to a car: Make, year and price. The file is memory-mapped for I/O efficiency. The file consists of 5 records initially along with the count of the records. The count of the records is placed at the beginning of the file. You can use the following structure for keeping records in the file:

```
struct car{
      char make[16];
      int year;
      int price;
};
```

Using the menu, the user can opt to add, edit and display records to/from carinfo.bin. Your program will create a **thread** to perform the required task (add, replace, display).

addrecord - This routine's only argument is a record entry. The given record entry will be appended to carinfo.bin and the count for the records will be incremented as well.

replacerecord - This routine will take two arguments: a record and an index. The record with the given index in the file /memory mapped region will be replaced by the one given as an argument. You can assume that the index given is always valid / in range. (See sample output)

displayall - This routine takes no arguments. It prints all records present in memory mapped region / file.

On exit, you have to unmap the region and close any open files before exiting.

Neither of the threads will return an error.

Note:
User is allowed to add as many records as he/she wants. The maximum number of records will never exceed 128.

Sample run

```
Enter 1 to add ; Enter 2 to replace; Enter 3 to display all; Enter 4 to exit: Enter choice: 2

Enter make:Pagani
Enter year: 2013
Enter price: 99999999
Enter index to replace: 2
Record 2 replaced successfully.

Enter 1 to add ; Enter 2 to replace; Enter 3 to display all; Enter 4 to exit: Enter choice: 4
Exiting.
```

Sample (text) view of the memory-mapped region before running.	**Sample (text) view of the memory-mapped region after sample run.**
5 BAC 2015 5000000 Vauxhaull 2010 750000 Nissan 2014 1200000 Ariel Atom 2015 2000000 Caterham 1999 100000	5 BAC 2015 5000000 Vauxhaull 2010 750000 Pagani 2013 99999999 Ariel Atom 2015 2000000 Caterham 1999 100000

```
/* local date and time example in C/C++ */
// For C
#include <stdio.h>
#include <time.h>
// For C++
#include<iostream>
using namespace std;
int main ()
{
```

```
  time_t rawtime;
  struct tm * timeinfo;

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );
// for C use
  printf ( "Current local time and date: %s", asctime (timeinfo) );
// for C++ use
  cout <<  "Current local time and date: " << asctime (timeinfo) ;
  return 0;
}
```