CSCD320 Programming Assignment, Winter 2012
Eastern Washington University, Cheney, Washington.

Total: 100 points.
Due: 11:59pm, March 18, 2011 (Sunday)

Please follow these rules carefully:

1. Write your name and EWUID on **EVERY** page of your project report and **EVERY** source code file. If you do not want to release your source code, you will need to get my written approval.
2. Verbal discussions with classmates are encouraged, but each student must independently work on his/her own project, write his/her own project report, without referring to anybody else's solution.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Blackboard system). Send in whatever you have by the deadline.
4. Submission will include two parts, which can be bundled into one zip file.
   (a) Project report.
   (b) Source code files, accompanied by a readme file that includes the instructions on how to compile and run your program.

   Your whole project submission must be named as: **firstname_lastname_EWUID_cscd320_prog.zip**, in which your report PDF file must be named as: **firstname_lastname_EWUID_cscd320_prog.pdf**
   (a) We use the underline '_' not the dash '-'.
   (b) All letters are in the lower case including your name and the filename's extend.
   (c) If you have middle name(s), you don't have to put them into the submission's filename.
5. You project report must be computer typeset in the **PDF** format. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this assignment as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and are also used by nearly all the computer science and math professionals for paper writing.
6. Sharing any content of this assignment in any way with anyone who is not in this class of this quarter is NOT permitted.

Other notices:

1. Each problem is equally treated. Pick one problem that you like. You won't be given extra (resp. less) points if you choose a harder (resp. easier) problem. Your choice only depends on your interest and the challenges that you want to give to yourself.
2. You are free to use any programming language, but scripting languages are not encouraged and also you are not allowed to use library-provided subroutines which you are asked to implement. For example, you cannot use a library subroutine to sort a sequence of data that you are asked to sort.
3. The project report must include at least the following components in a professionally written language and structure: an abstract of your work; an introduction and motivation of your work; your overall algorithmic idea and detailed algorithms; your strategies for your implementation; the experimental study results of your program including any data/charts/figures/tables that show the performance of your program; any conclusive observations that you can make about you work; any future work this project may continue with.
4. Check out any professional computer science papers to see how a professional writing looks like.

1 GPS simulation

This project aims to build a simulation of GPS, which will interacts with a simulated vehicle. The program will mainly have two components: one is the GPS and the other is the vehicle, but you are free to add other components and features if you like.

- The input for the GPS is a map (a graph), a starting position(a vertex), and a destination(another vertex). The job of GPS is to compute a fastest route (shortest path) from the starting position to the destination on the given map. You can build your own notion of the "shortest path". For example, you can consider the combination of the following: the condition of the road, speed limit, distance, traffic load, highway or not highway, etc.
- The vehicle is to follow the route given by the GPS. You can control the vehicle's speed according to the speed limit of the road. The vehicle and GPS should run in parallel and communicate with each other continuously at any time during the trip. You may introduce some level of foolishness into the driver, so that he/she may have some probability of missing the right exit/entrance given by the GPS at an intersection. In that case, the vehicle should ask the GPS to quickly re-compute a fastest route from that position to the destination. Of course, the driver's probability of missing the route should be small (you can control it as a parameter); otherwise, the driver will almost never get to the destination if the map is large, and we don't want this to happen.
- The GPS and vehicle can be implemented as two threads (This is only my idea. You can think of any other better idea). These two threads run in parallel and communicate with each other. Especially when the vehicle reaches an intersection, it must communicate with GPS to get the right direction.

The program does not have to have a GUI. You can print the trace out on your computer's screen in an appropriate format, but I suggest you to design a flexible structure of your code, so that you can extend this program and plug in a GUI later when you take the GUI course, making your program fancy.

2 B-tree's implementation on hard drives

The B-tree data structure is designed for massive data indexing and searching on external memory (EM). The goal of this project is to obtain a loyal implementation of the B-tree data structure on the hard drive. However, the programming languages such as C/C++ and Java do not provide mechanisms that you can directly use to manage a tree which is stored on the hard drive. For example, the following example questions need to be answered in order to obtain the EM-based B-tree's implementation.

- What is the notion of pointer/references in the hard drive space ? How do you implemente the mechanism of pointers/references on the hard drive, so that you can visit the tree nodes on the hard drive by using the pointers.
- Once the notion of EM-based pointers/references is clear, how do you manage the tree nodes on the drive, so that the hard drive space is most efficiently used ? How do you save/delete a node on the hard drive, so that you are wasting the least amount of hard drive space ?

- How do you implement a clever buffer-cache mechanism for this B-tree implementation in order to optimize its performance ?
- You will need to implement all the relevant operations of the B-tree data structure that we have discussed in class.

The final goal of your program is to provide a clear and easy-to-use interface to the end users, so that the users can call your program to build the B-tree for their data. After the running of the program, the B-tree index will be saved on the hard drive, so that later on whenever the user wants to perform search/insertion/deletion operations, that B-tree on the hard drive can be used.

This is an open problem to me too. I have no standard answers to the questions that I have raised. I believe there is no textbook covering this topics either. Of course, the industry must have had a very well-tuned implementation of the B-tree data structure for hard drives, but just for our own fun, how will you approach this implementation if this challenge is given by your real boss ?

3 Sort massive data

This project is to continue and implement the idea that we have studied for the problem4 of homework4. I encourage you to use the external merge sort idea [1] to sort a massive data set which is too large to fit into the RAM. Read the homework4 problem to recall the problem.

The simple program (`datagen.c`) that I provide is for you to generate a 10G integers from the range of [0, 0x7fffffff]. You need to compile this C program and then run it, which will generate and save 10G integers into a file called `‘data.txt’`. Each integer occupies one line in `‘data.txt’`. Note that although the data that I am giving is not very massive in order not to crash your system, the size of the data file will be about 100GB because each integer will have several characters stored in the file.

Your job is to implement an external sorting algorithm to sort this 10G integers and save the result into another file called `‘data.sorted.txt’`. In the sorted file, each integer also occupies one line, but all the data are saved in the ascending order.

You are allowed to change the code of `datagen.c`, so that you can generate data sets of different size for your various experiments and testings.

Commands to compile and run `‘datagen.c’` on Linux/MacOS/CygWin:

```
> gcc -o datagen datagen.c    /*compile the program*/
> ./datagen           /*run the program to generate the 10G integers*/
```

4 Your own choice

This is encouraged but will need my written approval (like email confirmation) first.

[1] http://en.wikipedia.org/wiki/External_sorting