REQUIRED: Please solve all four problems below to pre-qualify for the position

We would like you to participate in a first-round technical pre-screening, which consists of four software tasks below. When completing problems, please use these guidelines:

- When solving the problems, please solve them in Python unless another language is specified (e.g. for Problem 1 we want Linux commands, and Problem #4 is SQL)
- You may not use references unless otherwise specified. You may not ask friends for help or otherwise seek help.
- You may not share or discuss the problems with anyone else
- Please include how long it took you to solve each problem.
- We do not expect everyone to finish everything, so if you spend more than 90 minutes on the test, you can stop there, add any additional thoughts you may have and send what you have already completed. Please solve the problems sequentially, however. Do not move on to Problem #4 unless you have solved Problem #3.
- Problems #1 and #2 should be clear indicators of whether or not you have the core linux and programming fundamentals to qualify for this job. If you struggle with either of these problems, do not waste your time on Problems #3 and #4.
- If you find any of the problems to be extraordinarily hard or think any will require extensive research and new learning to solve, this job is probably not a good fit for, so you do not need to waste your time trying to solve the other problems. In our experience, many people find Problem #3 to be challenging and often need to spend a few minutes remember what linear time is all about, but you should still be able to solve it.
- **WARNING: If you cheat, use references when you are not supposed to, talk to people about the problems, share the problems with friends, or otherwise do unethical things, it will be very obvious to us in future interviews. Suspected cheaters will be reported to their employers and added to the engineer blacklists we share with several other companies. We take cheating and fraudulent activities very seriously. We do not expect everyone to know everything so WHEN IN DOUBT, SKIP IT AND MOVE ON.**
- **With that said,** we understand that programmers may not always remember the exact syntax for a given function, so just relax and do your best. If you can't remember something or have doubts about the exact syntax, feel free to add a comment on that line to clarify your thoughts. If there is something you are kind of hacking out but see as suboptimal, feel free to add a # FIXME. Do not worry -- even the best of programmers do this. Relax and just give it an honest attempt.

Problem #1 - Linux Fundamentals

Using nothing but a text editor and no references whatsoever, name twenty commands that can be executed on the command line. Two such examples are *ls* and *vi*. Avoid the temptation to use google, man pages, shell auto-completion, and do not even type the command in shell to test them. If you cheat, it will be very obvious during subsequent interviews and we will report you to your employer and add you to our blacklist.
Do not send more than 5 or 10 minutes on this task.

Problem #2 - Programming Warm Up

Write a function that takes a string and return a boolean stating if a string starts and ends with an upper-case letter A-Z.

Rules:

- You may not do something trivial like calling is_uppercase().
- You may not use any references and may not use any development environment to run or test your code.
- Code efficiency does not matter, but it should be reasonably tidy and bug-free.

Problem #3 - Programming with algorithms and data structures

Write a function that takes an array of numbers and the length of the array. The array of numbers can be of any length with the numbers being any size, positive or negative. You will be evaluated on the efficiency of the solution and tidiness/quality of your code.

Your function should:

- Report the range of the numbers (i.e. min and max)
- Print to screen a list of all numbers within the range that are missing from the array
- Print the count of all numbers that appear 2 or more times within the array
- Include documentation in-code and follow other "best practices" that you would for production code

Rules:

- You may not use any helper functions such as `min(), max(), find(), filter()`, etc
- Your solution must run in linear time `O(N)`
- No sorting is allowed (because then it would not `O(N)`)
- Your solution must be memory efficient. Even if input is [-9999999999. 9999999999].
- Assume the compiler/interpreter is dumb. For the purpose of the task, assume the compile will do the wrong thing and not magically optimize your code if you do something questionable.
- You must explicitly identify the data structure as you use them. For example if you are use foo[], on the same include a comment saying if you plan for it to be a LinkedList, an Array, or a ??? data-structure
- You may use reference ONLY to clarify the basic syntax of a function and if you need a short refresher on Big-O notation and linear-time, but you may not search for the answer or use sites like StackOverflow where people "tell you how to do something". If you use any references please include links to them at the top of your program.

Example: If your function is given `[3, 1, -5, 3, 3, -5, 0, 1, 1, 3]` and the corresponded array length of 10, the program would output the following:

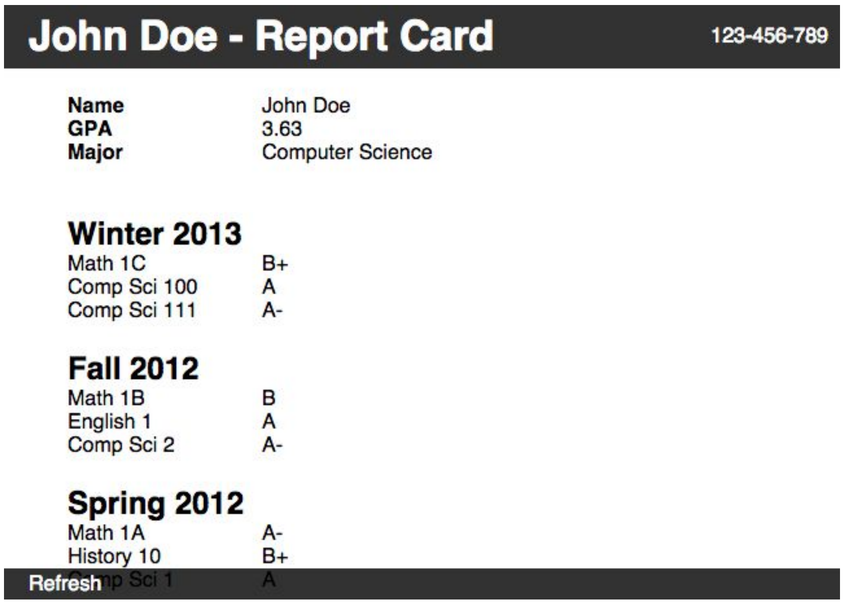
```
Range is -5 to 3

Missing Numbers:
-4
-3
-2
-1
2

Duplicate Numbers:
3 appears 4 times
1 appears 3 times
-5 appears 2 times
```

Problem #4 - Backend for a Webapp

Our goal is to build a web-based Report card system for a university that will look similar to the screenshot below. You may not use any references.



John Doe - Report Card 123-456-789

Name	John Doe
GPA	3.63
Major	Computer Science

Winter 2013

Math 1C	B+
Comp Sci 100	A
Comp Sci 111	A-

Fall 2012

Math 1B	B
English 1	A
Comp Sci 2	A-

Spring 2012

Math 1A	A-
History 10	B+
Comp Sci 1	A

Refresh

Clarifications:
- Math 1c, Comp Sci 100, etc represent class numbers. For example English 1 is the first English class a student takes. Comp Sci 111 might be a class on Computer Operating Systems while Comp Sci 2 might be a class that teach C++.
- GPA stands for Grade Point Average and is an average of letter grades to a numerical score. Letter grades correspond to number grades as follows:

GPA	Letter Grade	Percentage Grade		GPA	Letter Grade	Percentage Grade		GPA	Letter Grade	Percentage Grade
4.00	A	94-100		2.67	B-	80-83		1.33	D+	67-69
3.67	A-	90-93		2.33	C+	77-79		1.00	D	64-66
3.33	B+	87-89		2.00	C	74-76		0.67	D-	60-63
3.00	B	84-86		1.67	C-	70-73		0	F	0-65

Solve the problems below:
(a) Design a SQL Schema to store the data above. The SQL syntax for creating tables does not matter, but I do care about the tables, columns, data types for each, keys, relationships, etc
(b) Write the SQL Queries you would need to fetch the data from the database we designed in problem from (a). You do not need to worry about Python/PHP connections and can provide just the SQL Queries directly.
(c) If the database needed to be run for a university with billions of students, what are some ways you would optimize and scale the database? You do not need to implement any of this, but rather tell us some ideas at a high level.
(d) Extra Credit (Optional): build a small MVC app that lets users view their own Report Card