**MCS 253P Lab 4**

**Note from Google -- How to succeed: At Google, we believe in collaboration and sharing ideas. Most importantly, you'll need more information from the interviewer to analyze & answer the question to its full extent.**

**\* It’s OK to question your interviewer.**

**\* When asked to provide a solution, first define and frame the problem as you see it.**

**\* If you don't understand - ask for help or clarification.**

**\* If you need to assume something - verbally check it’s a correct assumption!**

**\* Describe how you want to tackle solving each part of the question.**

**\* Always let your interviewer know what you are thinking as he/she will be as interested in your process of thought as your solution. Also, if you're stuck, they may provide hints if they know what you're doing.**

**\* Finally, listen - don't miss a hint if your interviewer is trying to assist you!**

**What is Google looking for?:**

**"We are not simply looking for engineers to solve the problems they already know the answers to; we are interested in engineers who can work out the answers to questions they had not come across before." Interviewers will be looking at the approach to questions as much as the answer:**

**\* Does the candidate listen carefully and comprehend the question?**

**\* Are the correct questions asked before proceeding? (important!)**

**\* Is brute force used to solve a problem? (not good!)**

**\* Are things assumed without first checking? (not good!)**

**\* Are hints heard and heeded?**

**\* Is the candidate slow to comprehend / solve problems? (not good!)**

**\* Does the candidate enjoy finding multiple solutions before choosing the best one?**

**\* Are new ideas and methods of tackling a problem sought?**

**\* Is the candidate inventive and flexible in their solutions and open to new ideas?**

**\* Can questioning move up to more complex problem solving?**

**Google is keen to see really high quality, efficient, clear code without typing mistakes. Because all engineers (at every level) collaborate throughout the Google code base, with an efficient code review process, it’s essential that every engineer works at the same high standard.**

1. (20 pts) Write a C++ program to solve this popular job interview problem: Given an array input[] of n (where n >= 3) integers, construct an array output[] (of same size) such that output[i] is equal to the product of all the elements of input[] except input[i]. Solve it **without division operator and in O(n) time and space**. (**Note the simple and obvious solution is O() or uses division to remove the ith element or multiplication by the inverse which is the same as division**

Example: input = {10, 3, 5, 6, 2} output = {180, 600, 360, 300, 900}

Do not look at on-line solutions until after you solve it yourself. Your program should read a list (or vector) of N numbers from the standard input, one number per line, and will output the answer list (or vector) one product number per line. (SPOILER (means gives the answer away) TEXT IN WHITE: you will need one vector to store the input numbers, one to store the product of numbers below the ith, one to store the product of numbers above the ith. Then you can compute (and print) the answers by combining the values from above and below in a separate pass. END OF SPOILER)

1. Write a program to evaluate reverse polish expressions with signed 32 bit integers. Implement plus, minus, times, and divide. <http://en.wikipedia.org/wiki/Reverse_Polish_notation> Also implement the following commands

p - print out the value on top of stack,

s - swap top two items on the stack,

d - duplicate the top value of the stack

sample input (a list of expressions)

256 32767 2 \* 2 + s / p

-123456789 44 + 88 + 99 + 987654321 - 4 \* 2 / p

4 d d \* \* p

sample output (the evaluated expressions)

256

-2222221758

64