Velocix Programming Test

1.1 Aims

The aim of this exercise is to test your ability to write a working program in C++.

- We are looking for you to demonstrate that you can write good solid code, and debug it to get a good working solution. We consider it preferable to have a good solution that is finished and works rather than half-written code that tries to be too clever.

- Comments in your code will help us understand what you are trying to do in your program. But you can assume we are not stupid.

- We are looking for a pragmatic solution with the right trade-offs between performance and reliability given the time constraints.

- We are much more interested in allowing you to demonstrate your ability to design and implement data structures and algorithms to solve problems, so use of the STL or other similar container library is not permitted. The use of C++ I/O streams is permitted, the use of STL strings is not allowed.

- During the face to face interview you will be asked to explain the design of your solution and talk us through the code

- Most candidates take between 2 and 3 hours. Please do not spend longer than 4 hours on this task. In any case, please indicate approximately how long you spent writing your solution.

1.2 Languages and Tools

- The reference platform for the task is a Linux 64bit system running Ubuntu 10.04 LTS. If you do not have access to such a platform you should implement it on a Linux or Windows environment of your choosing, however, we will need to compile and assess your solution on Linux and we will only be able to make minor modifications to do so. Ensure you use portable libraries, and do not use Microsoft Project file formats to distribute your solution to us...

- The solution is to be written entirely in C++.

1.3 The Problem

Given a text file as an argument, your program will read the file, and output the 20 most frequently used words in the file in order, along with their frequency. The output should be similar to that of the following bash program:

```
#!/bin/bash
cat $1 | tr " .,();{}[]" "\n" | sort | grep -v "^$" | uniq -c |
sort -nr | head -20
```

Sample output, (this output is not from the reference text) :

```
      9 the
      5 you
      4 lessons
      4 can
      4 a
      3 Vim
      3 of
      3 is
      3 file
      2 users
      2 tutorial
      2 Tutor
      2 The
      2 that
      2 new
      2 make
      2 it
      2 in
      2 have
      2 for
```

The reference text for this test is the novel 'Moby Dick' available at http://www.gutenberg.org/etext/2489

Your program should also handle binary files (e.g. /boot/vmlinuz) without crashing, although the output can be undefined.