The University of the Witwatersrand, Johannesburg

School of Computer Science and Applied Mathematics

Data Structures and Algorithms
COMS2004

Ritesh Ajoodha

Teaching staff: Abiodun Modupe
Stuart Edwards
Ntokozo Mabena
Nabila Girka

Laboratory Notes 1

Student Name: _____

July 29, 2015

Contents

1 Definitions

Positive Integers The positive integers are the numbers 1, 2, 3, ..., sometimes called counting numbers or natural numbers, denoted Z^+ [Weisstein d].

Multiple A multiple of a number x is any quantity $y = nx$ with n an integer. If x and y are integers, then x is called a factor of y [Weisstein c].

Even Numbers An even number is any integer that can be divided by 2 with no remainder.

Pseudo-random Integers A set of values or elements that is statistically random, but it is derived from a known starting point and is typically repeated over and over. Pseudo-random numbers provide necessary values for processes that require randomness, such as creating test signals or for synchronizing sending and receiving devices in a spread spectrum transmission. It is

called "pseudo" random, because the algorithm can repeat the sequence, and the numbers are thus not entirely random [Company].

String A string of length n on an alphabet l of m characters is an arrangement of n not necessarily distinct symbols from l [Weisstein e].

Greatest Common Divisor (GCD) The greatest common divisor, sometimes also called the highest common divisor [Miller 1976], of two positive integers a and b is the largest divisor common to a and b. For example, GCD(3,5)=1, GCD(12,60)=12, and GCD(12,90)=6. The greatest common divisor GCD(a,b,c,...) can also be defined for three or more positive integers as the largest divisor shared by all of them. Two or more positive integers that have greatest common divisor 1 are said to be relatively prime to one another, often simply just referred to as being "relatively prime." [Weisstein b]

Chinese Remainder Theorem Let r and s be positive integers which are relatively prime and let a and b be any two integers. Then there is an integer N such that N≡ a (mod r) and N≡b (mod s) [Weisstein a].

2 Installing Xubuntu on your Home Computer

Many students have asked me to write a quick guide on how to install Xubuntu onto their home computer. Follow these instructions below:

(1) Download and install the latest version of Virtual Box using this website: "https://www.virtualbox.org/wiki/Downloads".

(2) Download the latest version of Xubuntu using this website: "http://xubuntu.org/getxubuntu/".

(3) Open Virtual Box

(4) Click New on the top left corner of the screen.

(5) When the pop-up screen appears: Name = "Xubuntu"; select type as "Linux"; and finally set version to be "Ubuntu" 32/64 bit depending on your system architecture. Click Next.

(6) In the "Memory size" window. Select the amount of RAM for your virtual machine. Use the recommended setting if you are unsure. Click next.

(7) In the "Hard drive" window. Check 'create a virtual hard drive now' and click "Create". For the "Hard drive file type", check "Virtualbox disk image", click next.

(8) In the "Storage on physical hard drive" window, select Dynamically allocated, click next.

(9) In the "File location and size" window. You are reqired to give a few GB of your hard drive to Xubuntu. I recommend using a minimum of 20GB of storage, finally click "Create".

(10) Click the "Settings" menu on the top left of the main window.

(11) Click "Storage" on the left pane.

(12) Under "Controller IDE" click "Empty".

(13) Select the CD icon after the text: "CD/DVD drives"

(14) Select "Choose a virtual CD/DVD Disk File", and select the Xubuntu .iso file you downloaded in step 2. Click Ok, Ok.

(15) Finally Click Start (Top Left), and follow the Xubuntu installation guide.

(16) You will be required to re-start Ubuntu on the Virtual Machine. Once the system restarts you will notice that Xubuntu can't go into full screen mode and that the system is quite slow.

(17) Once the System restarts. On the top window pane you will see a small menu containing: "Machine", "View", "Devices", and "Help". Click on "Devices"¿¿ "Insert Guest Additions CD image".

(18) On the Desktop, open the "VBOXADDITIONS" with the CD image icon. Finally click on 'autorun.sh', you will be required to enter your password that you chose during the Xubuntu installation.

(19) When you restart the system once more you should have a faster system that is able to go into full screen mode.

3 Xubuntu, Internet Access, and Files Storage

In the labs you'll be using **Xubuntu**. If your computer is running Windows, shut down Windows, open VirtualBox, and start the virtual machine called "general-linux-revision-1". For now, Xubuntu will automatically log in.

The computers in the New Science Stadium are not directly connected to the Internet. Instead they send requests to the CNS (Computer Network Services) proxy server which connects to the Internet for you. This proxy server requires a username and password, which is the same username and password used for your Wits email.

To browse the Internet you need to open Firefox. Firefox will now ask you for the login details for the CNS proxy. In the username box, enter **students\[your student number]**, example: students\44444. Your password should have been messaged to you confidentially. If not, try your ID-Number, unless you are an international student - in which case you should use your Passport Number.

If you have any problems with this username/password you need to go to CNS on the first floor of Senate House. Alternatively, you can email CNS at *ITStudentHelp@wits.ac.za.* Certain sites work without needing a proxy username and password. These are Google, and any Wits site such as the course site on Moodle: *http://courses.ms.wits.ac.za.* If your password doesn't work, you can still continue with this lab and sort out your CNS password later.

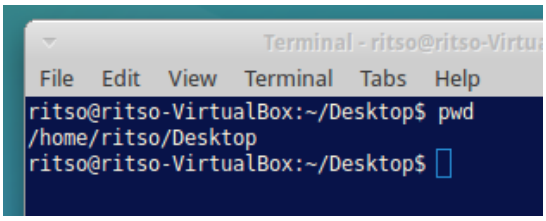
Please note that the New Science Stadium Laboratory was meant for Mathematical Science students and so we had to request access to use these labs. Thus, as soon as ICAM systems and class lists are finalized, some students may have issues getting into the building and using these computers.

We recommend that you make use of the **15GB** of cloud storage space provided for you on Google Drive to save files. You can access Google Drive when logging onto your Wits email via Firefox.

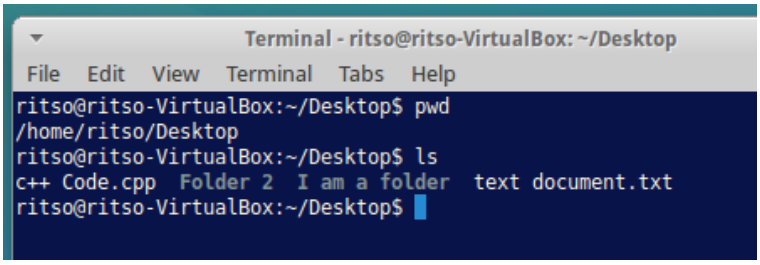
4 Using the Xubuntu Terminal

A basic understanding of Linux terminal commands can be very useful when programming using Linux based systems such as Xubuntu. We will quickly visit some of these commands in this section.

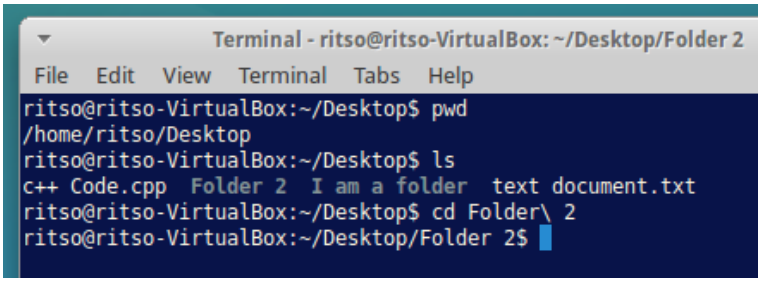
(1) To get the present working directory, type in 'pwd':



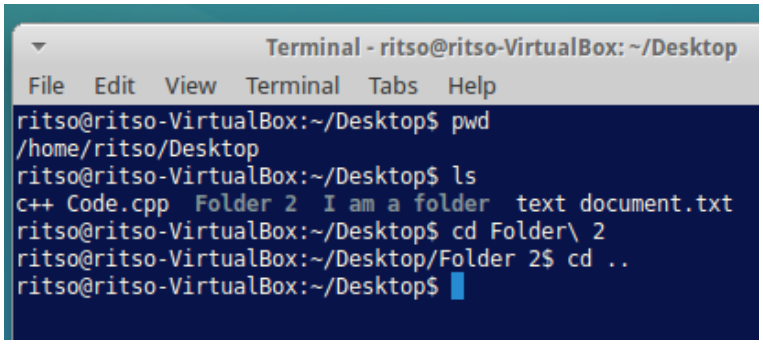
(2) To get a list of the files and folders that reside in the current working directory use the 'ls' command:



(3) To change directories use the 'cd' command followed by the name to the directory that you want to go to. Note that folder names that have spaces are specified using "\ " as shown below:

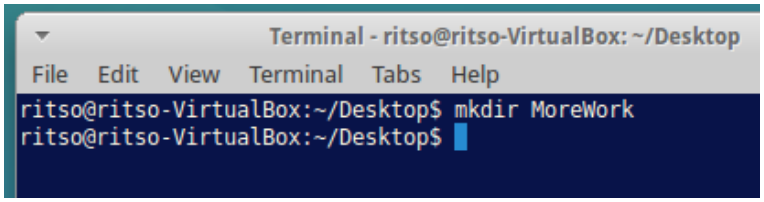


(4) The 'cd ..' command can be used to navigate back to the parent directory:

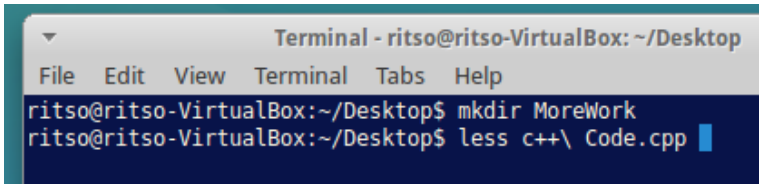


(5) You can clear the terminal using 'clear'.

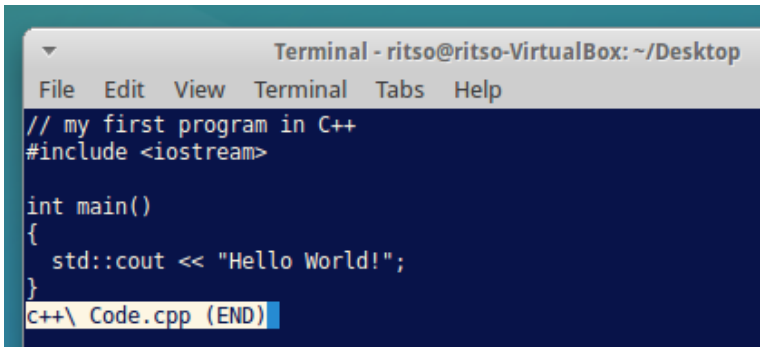
(6) If you need to make a directory, simple use the command 'mkdir' followed by the name of the directory you wish to create:



(7) If you want to display the contents of a file in the terminal. You can use the 'less' command followed by the name of the file:



Press enter:



Press "q" to exit the display. You cannot edit the file text in this display setting.

5 Compiling a C++ Program

In this lab you will learn how to create, compile and execute C++ programs. We will **Not** be using an Integrated Development Environment (IDE) to write and execute programs during lab sessions. Instead we will use a text editor (eg. gedit) to write code and terminal to compile and execute code.

(1) Right click on the desktop and create a folder, call this folder COMS2004Labs.

(2) Double click this folder.

(3) Right click and create another folder, call it Lab1.

(4) Double click this folder.

(5) Right click and under Create New Document, create an empty document called program.cpp

(6) Double click the file to open it. This starts gedit , a simple program for editing text. **(Do not use Emacs)**

(7) Type the following code into the file:

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

(8) Save the file.

(9) Now we will compile the program using the g++ compiler in the terminal. Right click on an empty space in the Lab1 folder and select the option 'Open in Terminal'. This will open a new terminal window with the present working directory set to the folder that your program is saved in.

(10) In the terminal window type in the following command and press enter to compile the program:

```
g++ -Wall program.cpp -o program.o
```

(11) The above command compiles the program and by using the -o option outputs an executable file called program.o. The -Wall option turns on all optional warnings which are desirable for normal code, this is helpful for debugging errors in your code.

(12) To execute the program type the following command in the terminal window and press enter:

```
./program.o
```

(13) The following output should now be displayed in the terminal window: Hello World!

(14) Note. Every time you make a change to a program you will need to compile it before executing it.

6 Submitting to the Online Marker

Now that you are able to compile and run programs on Xubuntu, you will need to submit your code to your instructor to be evaluated. This is done using the online marker. Do this by following the steps below:

(1) Use your browser to navigate to the following site:
http://courses.ms.wits.ac.za/moodle/login/index.php.

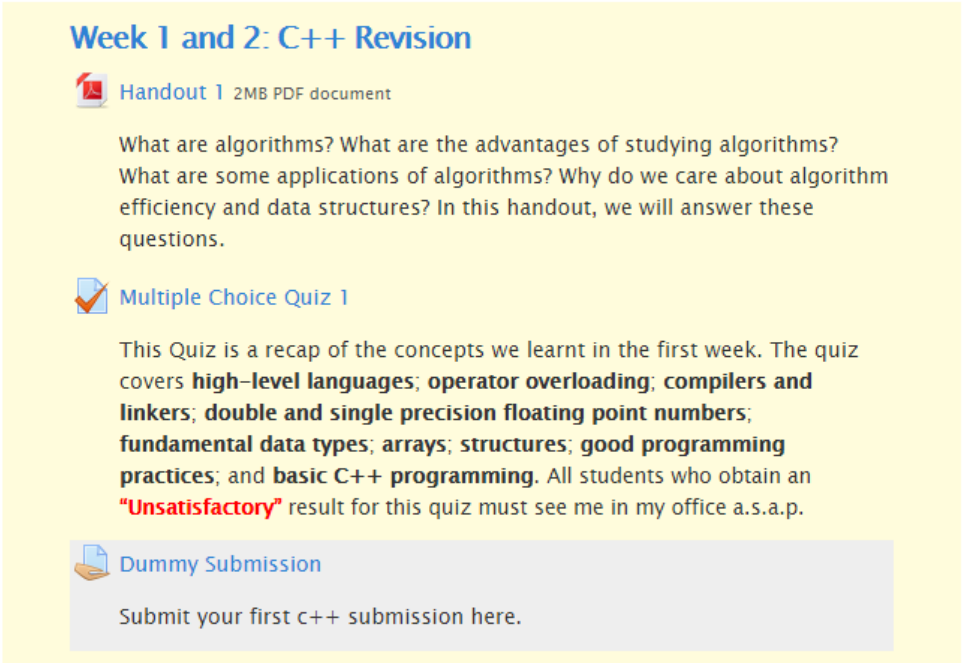
(2) Once youve logged on to Moodle click on 'CS2015' under 'School of Computer Science'.

(3) Click on "Data Structures and Algorithms by Ritesh Ajoodha". Be careful **NOT** to click on "Introduction to Data Structures & Algorithms (COMS1017) by Richard Klein" as this is a first year course offered by the School of Computer Science and is not applicable to you.

(4) Click 'Enrol me'

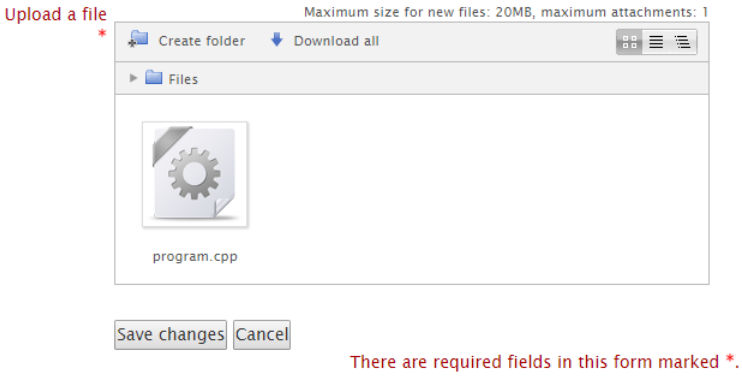
(5) This is the course website on Moodle. You can find out everything you need to know about the course from this website. This is the central component for all lab submissions and weekly quizzes. Please make sure you check with this website regularly to complete mandatory coursework submissions.

(6) Under Week 1 and 2: C++ Revision, click 'Dummy Submission'

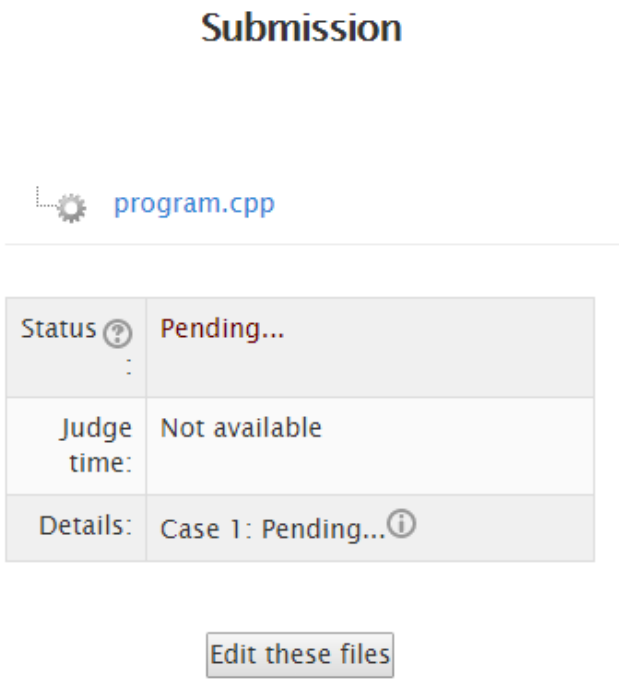


(7) Click "Upload files"

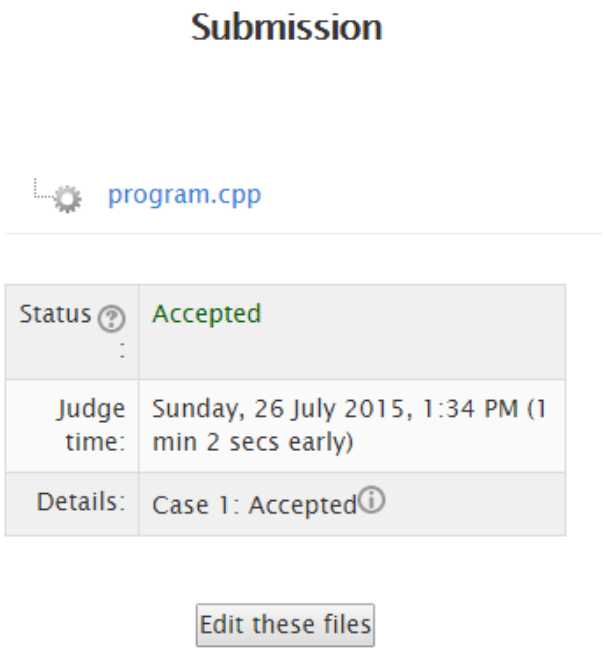
(8) Upload the C++ file you wish to submit, and click Save Changes



(9) Your submission should be 'Pending':



(10) Press **F5** to refresh the page, and you will know if your assignment is accepted or not:



(11) You can submit any of the Lab work as many times as you like. I am not concerned with how long it takes you to get your work Accepted, **as long as you eventually get it accepted**. Lab submissions, however, do have due dates. Always watch the submission deadlines!

References

[Company] The Computer Language Company. *Pseudo-random numbers.* Last visited on 19/7/2015.

[Miller 1976] Gary L Miller. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences*, 13(3):300–317, 1976.

[Weisstein a] Eric W. Weisstein. *"Chinese Remainder Theorem." From MathWorld–A Wolfram Web Resource.* Last visited on 19/7/2015.

[Weisstein b] Eric W. Weisstein. *"Greatest Common Divisor." From MathWorld–A Wolfram Web Resource.* Last visited on 19/7/2015.

[Weisstein c] Eric W. Weisstein. *"Multiple." From MathWorld–A Wolfram Web Resource.* Last visited on 19/7/2015.

[Weisstein d] Eric W. Weisstein. *"Positive Integer." From MathWorld–A Wolfram Web Resource.* Last visited on 19/7/2015.

[Weisstein e] Eric W. Weisstein. *"String." From MathWorld–A Wolfram Web Resource.* Last visited on 19/7/2015.

THE UNIVERSITY
OF THE WITWATERSRAND

Lecturer: **Ritesh Ajoodha**

Laboratory 1
Data Structures and Algorithms
July 22, 2015
Due: August 5, 2015

These questions are adapted from the text book: DATA STRUCTURES AND ALOGORITHMS IN C++ by Michael Goodrich, *Roberto* Tamassia, *and David* Mount.

Question R 1.7

Write a short C++ program that takes an integer n and returns the sum of all the positive integers smaller than n.

HINT: *The easy solution uses a loop, but there is also a formula for this, which is discussed in Chapter 4.*

```
Sample I/O 1:
Input: 7
Output:21

Sample I/O 2:
Input: 21
Output:210
```

Question R 1.8

Write a short C++ program, isMultiple, that takes two positive long values, n and m, and returns true if and only if n is a multiple of m, that is, $n = m.i$ for some integer i.

HINT: *The modulus operator could be useful here.*

```
Sample I/O 1:
Input:  353644199316
        6548966654
Output: true

Sample I/O 2:
Input:  353649874199316
        6544568966654
Output: false
```

Question R 1.9

Write a C++ function *printArray(A, m, n)* that prints an $m \times n$ two dimensional array A of integers, declared to be "int** A," to the standard output. Each of the m rows should appear on a separate line. Your input will be as follows with the first two values being the number of rows and columns respectively.

HINT: *This is best done with one for-loop nested within another.*

```
Sample I/O 1:
Input: 3
       2
       1
       2
       3
       4
       5
       6

Output: 1 2
        3 4
        5 6

Sample I/O 2:
Input: 4
       2
       591
       751
       946
       134
       981
       127
       365
       744
       987

Output: 591 751 946
        134 981 127
        365 744 987
```

Question R 1.19

Write a short C++ program, *isTwoPower*, that takes an **int** *i* and returns **true** if and only if *i* is a **power of 2**. Do not use multiplication or division, however.

HINT: *Consider the binary representations of i and i - 1.*

```
Sample I/O 1:
Input:  1048576
Output: true

Sample I/O 2:
Input:  262140
Output: false
```

Question R 1.21

Write a short C++ program that takes an integer n and returns the sum of all the odd integers smaller than n.

HINT: *The easy thing to do is to write a loop.*

```
Sample I/O 1:
Input:  7
Output: 9

Sample I/O 2:
Input:  618432
Output: 1125254144
```

Question R 1.22

Write a short C++ program that takes a positive **double** value x and returns the number of times we can divide x by 2 before we get a number less than 2.

HINT: *Use a while loop.*

```
Sample I/O 1:
Input:  8.65
Output: 3

Sample I/O 2:
Input:  9173.54
Output: 13
```

Question C 1.3

Write a C++ program that takes an STL vector of **int** values and determines if all the numbers are different from each other (that is, they are distinct). The first line will be the size of the vector followed by actual elements.

HINT: *The simple solution just checks each number against every other one, but we will discuss better solutions later in the book. But make sure you don't compare a number to itself.*

```
Sample I/O 1:
Input:  6
        5
        9
        8
        7
        6
        4
Output: true

Sample I/O 2:
Input:  6
        5
```

```
        9
        8
        7
        8
        4

Output: false
```

Question C 1.4

Write a C++ function that takes an STL vector of **int** values and prints all the odd values in the vector. The first line will be the size of the vector followed by actual elements.

HINT: *The modulus operator could be useful here.*

```
Sample I/O 1:
Input: 6
       5
       9
       8
       7
       6
       4

Output:5
       9
       7


Sample I/O 2:
Input: 5
       8
       9
       8
       7
       3

Output:9
       7
       3
```

☆ Question C 1.5

Write a C++ program that takes an array containing the set of all integers in the range 1 to 52 and shuffles it into random order. Use the built-in function rand, which returns a pseudo-random integer each time it is called. Your function should output each possible order with equal probability.

HINT: *Consider using swaps to reshuffle the array one entry at a time.*

☆ Question C 1.6

Write a short C++ program that outputs all possible strings formed by using each of the characters 'a', 'b', 'c', 'd', 'e', and 'f' exactly once.

HINT: *There are many solutions. If you know about recursion, the easiest solution uses this technique. Otherwise, consider using an array to hold solutions. If this still seems to hard, then consider using six nested loops (but avoid repeating characters and make sure you allow for all string lengths).*

☆ Question C 1.7

Write a short C++ program that takes all the lines input to standard input and writes them to standard output in reverse order. That is, each line is output in the correct order, but the ordering of the lines is reversed.

HINT: *Use a (big) STL vector to store all the lines.*

```
Sample I/O 1:
Input:    Once upon a midnight dreary, while I pondered, weak and weary,
          Over many a quaint and curious volume of forgotten lore
          While I nodded, nearly napping, suddenly there came a tapping,
          As of some one gently rapping, rapping at my chamber door.
          "'Tis some visiter," I muttered, "tapping at my chamber door
          Only this and nothing more."

Output:   Only this and nothing more."
          "'Tis some visiter," I muttered, "tapping at my chamber door
          As of some one gently rapping, rapping at my chamber door.
          While I nodded, nearly napping, suddenly there came a tapping,
          Over many a quaint and curious volume of forgotten lore
          Once upon a midnight dreary, while I pondered, weak and weary,

Sample I/O 2:
Input:    Shall I compare thee to a summer's day?
          Thou art more lovely and more temperate:
          Rough winds do shake the darling buds of May,
          And summer's lease hath all too short a date:
          Sometime too hot the eye of heaven shines,
          And often is his gold complexion dimm'd;
          And every fair from fair sometime declines,
          By chance or nature's changing course untrimm'd;
          But thy eternal summer shall not fade
          Nor lose possession of that fair thou owest;
          Nor shall Death brag thou wander'st in his shade,
          When in eternal lines to time thou growest:
          So long as men can breathe or eyes can see,
          So long lives this and this gives life to thee.

Output:   So long lives this, and this gives life to thee.
          So long as men can breathe, or eyes can see,
          When in eternal lines to Time thou grow'st.
          Nor shall death brag thou wand'rest in his shade,
```

```
Nor lose possession of that fair thou ow'st,
But thy eternal summer shall not fade,
By chance, or nature's changing course, untrimmed;
And every fair from fair sometime declines,
And often is his gold complexion dimmed;
Sometime too hot the eye of heaven shines,
And summer's lease hath all too short a date.
Rough winds do shake the darling buds of May,
Thou art more lovely and more temperate.
Shall I compare thee to a summer's day?
```

Question C 1.8

Write a short C++ program that takes two arguments of type STL vector<double>, a and b, and returns the element-by-element product of a and b. That is, it returns a vector c of the same length such that c[i] = a[i].b[i].

HINT: *You may assume that a and b are of equal size. Create a vector c having this same size.*

```
Sample I/O 1:
Input:  7 5 6 4 9 8 2 5 3 6
        7 6 4 5 1 2 6 4 9 8

Output: 49 30 24 20 9 16 12 20 27 48

Sample I/O 2:
Input:  1 2 3 4 5 6 7 8 9
        9 8 7 6 5 4 3 2 1

Output: 9 16 21 24 25 24 21 16 9
```

Question C 1.10

Write an efficient C++ function that takes any integer value i and returns 2^i, as a **long** value. Your function should **not** multiply 2 by itself i times; there are much faster ways of computing 2^i.

HINT: *Consider performing repeated squaring, starting with 2. How can we compute 2i from all of these perfect squares,* 2, 2^2, 2^4, 2^8, *etc.*

```
Sample I/O 1:
Input:  8
Output: 256

Sample I/O 2:
Input:  30
Output: 1073741824
```

Question C 1.11

The **greatest common divisor**, or GCD, of two positive integers n and m is the largest number j, such that n and m are both multiples of j. Euclid proposed a simple algorithm for computing GCD(n,m), where n > m, which is based on a concept known as the Chinese Remainder Theorem. The main idea of the algorithm is to repeatedly perform modulo computations of consecutive pairs of the sequence that starts (n,m, . . .), until reaching zero. The last nonzero number in this sequence is the GCD of n and m. For example, for n = 80,844 and m = 25,320, the sequence is as follows:

$$\begin{aligned} 80{,}844 \bmod 25{,}320 &= 4{,}884 \\ 25{,}320 \bmod 4{,}884 &= 900 \\ 4{,}884 \bmod 900 &= 384 \\ 900 \bmod 384 &= 132 \\ 384 \bmod 132 &= 120 \\ 132 \bmod 120 &= 12 \\ 120 \bmod 12 &= 0 \end{aligned}$$

So, GCD of 80,844 and 25,320 is 12. Write a short C++ function to compute GCD(n,m) for two integers n and m.

HINT: *Remember that x % y operator returns the remainder of the integer division x/y.*

```
Sample I/O 1:
Input:  143528
        8159088
Output: 56

Sample I/O 2:
Input:  86698832
        3215608
Output: 88
```

END OF LABORATORY 1