1 Instructions

- You may work with *one partner as a team of two programmers* on this lab project if you wish or you may work alone.
- If you work with a partner, before you submit your solution, ensure that *both you and your partner belong to the same Canvas group*. You can find the groups in Canvas by navigating to the *People* page and then clicking on the tab labeled *Project Groups*.
- If you intend to work with a partner, find an empty group and add yourself to the group and ask your partner to do likewise. You and your partner may continue to belong to the same *Project Group n* to the end of the term if you wish. Of course, you may always leave a group and join another group, so the partner you work with now does not have to remain your partner.
- Consult §5 for important submission instructions.

2 Lab Project Objectives

After completing this assignment the student should be able to,

- Complete all of the objectives of the previous lab projects.
- Write programs using **while** and **for** loops.
- Write programs consisting of functions defined in multiple source code files.
- Write function prototypes in header files and include header files when required.
- Use the **static** reserved words to specify private identifiers in source code files.

3 Prelab Exercises

3.1 Prelab Exercise 1: Create Replit.com Project and Add *main.cpp* to the Project

You may use any C++ development environment you wish to complete the project. If you wish to use an online C++ IDE so you do not have to install a C++ IDE on your personal computer, we recommend that you use Replit.com.

If you have not already done so, before your lab session, create a free Repl.it account. After doing that, create a new C++ project and name the project *lab08* (Note: if you do not have an account and are logged in as @anonymous, you will not be able to rename the project from the default random project name that was assigned to it when the project was created.) Then upload all the source code files found in the Lab Project 8 zip archive, overwriting the *main.cpp* file that was added to the project when you created it.

Before proceeding to the next prelab exercise, edit the header comment block of each source code file and add your information to the AUTHOR1: comment. If you work with a partner, add the AUTHOR2: comment including his or her information. Remember that only one member of the team has to upload the submission zip archive to Canvas but you will each earn the same score as long as you follow the instructions in this section and in §5.

For help using Replit, click the ? button in the lower-right corner of the IDE window. If you require more assistance, ask your lab TA or a UGTA for help in your lab session. Note that when when you are finished with the lab project, you may download your project files in a zip archive: Click the three dots symbol in the title bar of the *File Pane* and select *Download as Zip* from the menu. This will download a zip archive named *lab08.zip* (assuming you named your project *Lab 7* as requested).

Note: The downloaded zip archive named *lab08.zip* is **not** the zip archive you are required to upload to Canvas for grading. Please consult §5 for that information.

3.2 Prelab Exercise 2: Write a Vary Loop Using a For Loop

Edit the *vary_for1*() function in prelab.cpp so when called, it will print the integers -1, -2, -3, ..., -25 inside a pair of brackets on the output window using a **for loop**. The output shall be:

```
[-1□-2□-3□...□-23□-24□-25□]
```

where ... represent integers which are not shown here and the box symbol represents the space character. Note that a space character is output after each integer, including the last integer. Print one blank line before returning from the function.

3.3 Prelab Exercise 3: Write a Vary Loop Using a For Loop

Edit the *vary_for2*() function in prelab.cpp so when called, it will print the integers -1, -2, -3, ..., -25 inside a pair of brackets on the output window using a **for loop**. The output shall be:

```
[-1□-2□-3□...□-23□-24□-25]
```

where ... represent missing integers and the box symbol represents the space character. Note that a space character is output after each integer, *except* the last integer. This will require you to determine a way to print a space after each of the first 24 integers but not after the 25th integer. Print one blank line before returning from the function.

3.4 Prelab Exercise 4: Write a Vary Loop Using a While Loop

Edit the *vary_while*() function in prelab.cpp so when called, it will print the integers 100, 97, 94, ..., 4, 1 inside a pair of brackets on the output window using a **while loop**. The output shall be:

```
[100□97□94□...□4□1□]
```

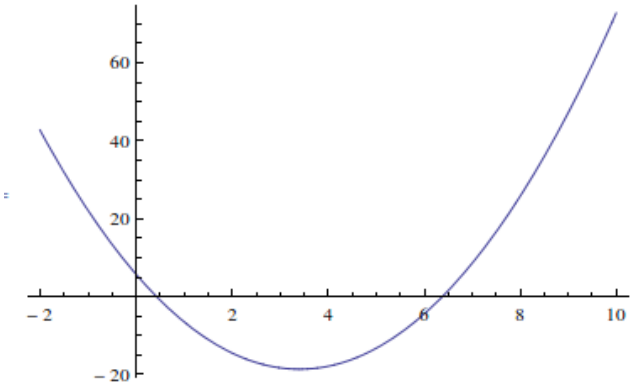
where ... represent missing integers which are not shown here and the box symbol represents the space character. Note that a space character is output after each integer, including the last integer. Print one blank line before returning from the function.

3.5 Prelab Exercise 5: Read Software Requirements

Read §4 *Software Requirements* of this lab project document which describes what the lab project program will do, i.e., the **software requirements**. Then come back here and complete the remaining prelab exercises.

3.6 Prelab Exercise 6: Background: Newton's Method

[Ref: Wikipedia: Newton's Method] **Newton's method** is a mathematical algorithm that can be used to find the roots of a real-valued function, i.e., the roots are real numbers. For example, consider the 2nd-degree polynomial function $p(x) = 2.1x^2 - 14.3x + 5.7$ and its plot,



Clearly $p(x)$ has two real roots: the first one, $root_1$, is around 0.45 and the second one, $root_2$, is near 6.5. Since $p(x)$ is a second degree polynomial equation, i.e., a quadratic, we could use the well-known quadratic formula to find the roots, but we could also find the roots using Newton's method.

Newton's method is an **iterative method**, i.e., it involves a loop. Here is the pseudocode for Newton's method which finds a root of $f(x)$ where $f(x)$ is any real-valued function,

function *newtons_method* ($f(x)$ is a real-valued function, ε is a **double**) → **double**
    $x_i \leftarrow$ guess a value for the root we are trying to find
    $x_{i+1} \leftarrow x_i - f(x_i) \,/\, f'(x_i)$ -- evaluate $f(x)$ at x_i and evaluate the first derivative, $f'(x)$, at x_i
    while $|x_{i+1} - x_i| \geq \varepsilon$ **do**
        $x_i \leftarrow x_{i+1}$
        $x_{i+1} \leftarrow x_i - f(x_i) \,/\, f'(x_i)$
    end while
    return x_{i+1}
end function

The parameter ε is a small number which controls how accurate the returned value is. For example, if we desire the returned value to be correct to three digits after the decimal point, then we would pass $10^{-4} = 0.0001$ as ε. If we passed $10^{-9} = 0.000000001$ for ε, the returned value would be accurate to at least eight digits after the decimal point. In general, if ε is 10^{-p}, the accuracy will be p - 1 correct digits after the decimal point. The smaller ε is, the more accurate the result, but this accuracy comes at the expense of performing more iterations to find the root, i.e., it takes longer. Note that in C++, we can represent real numbers in **exponential notation**, which is similar to scientific notation. For example, 123.456×10^{-78} would be represented as 123.456E-78, where the E represents "times 10 to the" and the exponent is written after the E. Consequently, if we wished to use 10^{-16} for ε, we would write 1E-16.

Note that Newton's method starts by initializing x_i to some value that is basically a guess for where we think the root is. The more accurate this guess, the more quickly Newton's method will converge to the correct answer. For a function with more than one root, e.g., our $p(x)$ function from above, this initial guess will affect which root (the one near 0.45 or the one near 6.5) is found. For example, if x_i were initialized to 0 then Newton's method is going to converge on the root near 0.45. On the other hand, if we guessed 100 for x_i then it will converge to the root near 6.5.

The loop iterates until the absolute value of the difference between the prior value of x (x_i) and the current value of x (x_{i+1}) is less than ε. Each time through the loop, a new value (x_{i+1}) for the root we are searching for is calculated by subtracting from the prior value (x_i) the quotient of dividing f(x) evaluated at x_i by the first derivative of $f(x)$ evaluated at x_i. (If you want to understand the mathematics behind Newton's method, i.e., why it works, refer to the Wikipedia page referenced above or any calculus text book). In my day, Newton's method was learned in Calculus II.

We are going to use Newton's method in this lab project to write a function that computes and returns the a^{th} root of a real number n, i.e.,

$$\sqrt[a]{n}$$

We will discuss how the algorithm works, using the square root of $n = 17$ ($a = 2$) as the example. If

$$x = \sqrt{n}$$

then,

$$x^2 = n$$

which we can write as,

$$x^2 - n = 0$$

If we define $f(x) = x^2$ - n, then we can use Newton's method to find the positive root of $f(x)$, which will be the square root of n. For example, let's trace *newtons_method*() for $f(x) = x^2$ - 17 and ε = 1E-4 = 0.0001, i.e., we are going to find the square root of $n = 17$ with at least three digits of accuracy after the decimal point. Note that $f'(x) = 2x$. Let's use n / 2 as our starting guess because the square root of n will be between 1 and n / 2 (accurate digits of x_{i+1} are underlined),

x_i	x_{i+1}	$\lvert x_{i+1} - x_i \rvert$	
8.5000000000000000	5.2500000000000000	3.2500000000000000	Difference is > ε so keep looping
5.2500000000000000	4.2440476190476186	1.0059523809523814	Difference is > ε so keep looping
4.2440476190476186	4.1248288586121689	0.1192187604354498	Difference is > ε so keep looping
4.1248288586121689	4.1231059855758616	0.0017228730363072	Difference is > ε so keep looping
4.1231059855758616	4.1231056256176766	0.0000003599581850	$\lvert x_{i+1} - x_i \rvert <$ ε = 0.0001 so return x_{i+1}

Notice how each new value of x_{i+1} is closer to the root than the previous value x_i. If Newton's method is going to converge, the absolute value of the difference of x_{i+1} and x_i will become smaller and smaller, until it is less than ε (in theory, the difference will go to 0 but the **double** data type cannot represent real numbers with more than 15-16 digits of accuracy). Consequently, *newtons_method*() would return 4.1231056256176766 (the underlined digits are all accurate) for the square root of 17. The correct value to 25-digits is 4.1231056256176605498214 10... (the correct value is a non-terminating fraction, i.e., it goes on forever) and the absolute value of the difference between the correct value to 25-digits and what *newtons_method*() returns is approximately 1.59872×10^{-14}, which is our error. Therefore, *newtons_method*() returned a value that is correct to |-14| - 1 = 13 digits after the decimal point *and* it only took four iterations of the while loop to do it. In

general, to find the a^{th} root of n, $a \geq 2$, we define $f(x) = x^a - n$ where $f'(x) = ax^{a-1}$. The pseudocode for our function (named *compute_ath_root*) is,

function *compute_ath_root*($n \geq 0$ is a **double**, $a \geq 2$ is an **integer,** ε is a **double**) → **double**
    $x\_i \leftarrow n / 2$ -- initial guess
    $x\_i\_plus\_1 \leftarrow x\_i - (x\_i^a - n) / (a \cdot x\_i^{a-1})$
    while $|x\_i\_plus\_1 - x\_i| \geq \varepsilon$ **do** -- notice this is a sentinel loop with the sentinel being a value less than ε
        $x\_i \leftarrow x\_i\_plus\_1$
        $x\_i\_plus\_1 \leftarrow x\_i - (x\_i^a - n) / (a \cdot x\_i^{a-1})$
    end while
    return *x_i_plus_1*
end function

We have to make a guess for the initial value of x_i so we will use n divided by 2. I believe, but have not verified, that this initial value will cause Newton's Method to always converge to the a^{th} root.

3.7 Prelab Exercise 7: Study the Software Design

The software design is documented as pseudocode in the source code files. Study the pseudocode.

4 Software Requirements

The program shall display the *Lab 8 Main Menu*:

```
-----------------------------------------
Lab 8 Main Menu: What would you like to do?
-----------------------------------------
1. Run the prelab exercises.
2. Compute the a-th root of a real number.
3. Quit the program.
Your Choice [1-3]?
```

If the user selects menu item 1, then the *Prelab Exercises Menu* will be displayed:

```
-------------------------------------------------
Prelab Exercises Menu: What would you like to do?
-------------------------------------------------
1. Prelab Exercise 2: Vary Using For Loop 1
2. Prelab Exercise 3: Vary Using For Loop 2
3. Prelab Exercise 4: Vary Using While Loop
4. Return to the Main Menu
Your Choice [1-4]?
```

If the user selects menu item 1, then the *vary_for1*() function of Prelab Exercise 2 shall be performed. After it returns, the Prelab Exercises Menu will be displayed again, prompting the user to select a menu item. If the user selects menu item 2, then the *vary_for2*() function of Prelab Exercise 3 shall be performed and after it finishes, the Prelab Exercises Menu will be displayed. If the user selects menu item 3, then the *vary_while*() function of Prelab Exercise 4 shall be performed and after the function returns, the menu shall be redisplayed. If the user selects menu item 4, then they shall be returned to the Lab 8 Main Menu, which will be displayed again.

If the user selects menu item 2 in the Lab 8 Main Menu, then *process_ath_menu*() in main.cpp shall be called. It will prompt the user to enter a real number for n and an integer for a, as described in Prelab Exercise 6.

```
Your choice is to compute the a-th root of n.
Enter n (>= 0): 13512
Enter a (>= 2): 5
```

The program will then call *compute_ath_root*() in main.cpp, which shall compute the a^{th} root of n and shall return it back to *process_ath_root*() which will display the root:

```
The 5-th root of 13512.0000000000000000 is 6.7010664923861052
```

and then return back to *run*() which will immediately display the Lab 8 Main Menu again. The user can terminate the program by selecting Lab 8 Main Menu item 3.

4.1 Additional Programming Requirements

1. Add a **comment header block** at the top of each source code file with your name, ASURITE ID, email address, the lab number in the AUTHOR1 comment, and include your lab date/time and your lab TA. If you worked with a partner, include your partner's name, ASURITE ID, and email address in the AUTHOR2 comment.
2. Carefully **format** your code and follow the **indentation** of the text as shown in the example programs of the textbook.

5 Submission Instructions

- For help information about how to use Canvas, please visit the Canvas Student Guide.
- You are permitted to work in a team of 2 students. In Canvas, we have created multiple groups, as described in §1 Instructions. If you wish to work with a partner, please follow the instructions in that section to add your partner and yourself to the same group. If you require assistance, please consult the help documentation about how to join a group in this page of the Canvas Student Guide.
- When you are ready to submit your Lab 8 project, begin by using the file explorer program of your operating system to create a new empty folder named lab08-*asuriteid* where *asuriteid* is your ASURITE user id that you use to log in to MyASU (e.g., mine is *kburger2* so my folder would be named lab08-kburger2). If you worked with a partner in a team of two on the project, put both of your ASURITE id's in the name of the new folder, e.g., lab08-*asuriteid1-asuriteid2.*
- Download the source code files *main.cpp*, *get.cpp*, *get.hpp*, *prelab.cpp*, and *prelab.hpp* from Replit and copy them to your lab08-*asuriteid* folder.
- Compress the lab08-*asuriteid* folder creating a zip archive named lab08-*asuriteid*.zip or lab08-*asuriteid1-asuriteid2*.zip.
- Submit the lab08-*asuriteid*.zip or lab08-*asuriteid1-asuriteid2*.zip archive to Canvas by the deadline using the *L8: Lab Project 8* submission page.
- If you worked with a partner, the first student who adds him- or herself to the group will be designated as the *group leader*. If you worked in a team, the group leader shall upload the solution zip archive to Canvas. If you worked alone, then obviously, you shall submit it.
- Once you submit a file in Canvas for grading, you can download the zip archive and verify that it is not corrupted and that it is named correctly and contains the correct source code files. Do this as soon as you submit!
- If your program does not compile due to syntax errors or if it does not run correctly because one or more of the software requirements are violated (i.e., the code contains one or more bugs), then upload what you have completed for grading anyway because you will generally receive partial credit points for effort.
- The submission deadline is **11:59pm Sat 10 Apr**. We shall permit lab submissions, up to 48 hours after the deadline, or in other words, you must submit before the 48-hour late submission deadline of **11:59pm Mon 12 Apr**. The penalty for a one day late submission is 10% of the project points (-0.5 pts). For a two day late submission, the penalty is 20% or -1 pt.
- The Canvas submission link will become unavailable at the end of the day on Mon 12 Apr and you will not be able to submit your project for grading after that time. We do not accept emailed submissions or submissions that are more than 48 hours later. Do not ask.
- Note that we may employ an automated grading script to grade your lab projects. This script will depend on you naming the submission files as requested in the document. **Failure to follow this naming convention may result in a score of 0 being assigned for your lab project, even if it builds and runs correctly**.
- To ensure you each receive the same score, it is imperative that you follow these instructions and those in §5 Submission Information, i.e., put both of your ASURITE ID's in the submission archive filename and type both of your names, ASURITE ID's, and email addresses in the AUTHOR1: and AUTHOR2: lines of the header comment block for each source code file.
- Consult the online syllabus for the academic integrity policies.

6 Grading Rubric

To be completed later.

