If this lab is an Individual assignment, you must do all coded programs on your own. You may ask others for help on the language syntax, but you must organize and present your own logical solution to the problem. No lab is complete until the student submits the signed pledge form associated with that lab. I realize that no coded programs will be graded until I turn in the sign & pledge form associated with that program; any late penalties will continue to compound until the pledge form is submitted.

If this lab is a team assignment, both team members may share logic as they program side by side on their own computers. Each person must type all of his/her own code as part of the learning process. Team assignments are never to be “You do this portion and I’ll do that portion” or “You do this lab and I’ll do the next lab”.

Some of the lab assignments will have short answer questions. These short answer questions will be spot checked and graded for completion, but not checked for accuracy. I encourage you to get into a study group to help each other prepare for exams, quizzes, etc. Once these labs are graded and returned, I encourage you to compare answers with another class member, in your study group, who has also had the lab graded and returned.

I/We realize that the penalty for turning in work that is not my own, or assisting others in doing so, can range from an "F" in the class to dismissal from Trinity University. I realize that it is a violation of academic integrity to share any portion of this lab with any person (outside my 2320 team & professor)!

Print Name _________________________________________ Time Required = ______.____ Hrs.
Signature _______________________________________________________________ (pledged)

Review → OOP-1 Review Of C & Memory Management
Individual Assignment
15 Points

Short Answer Questions
Electronic Solutions Of Short Answer Questions Will Not Be Accepted. Print A Copy Of Short Answer Question & Write Answers On Printed Copy

Read The Course Outline.

1] ______ This semester, CSCI 2320 will have _?_ exams.

2] ______ {T/F} This semester, CSCI 2320 will have quizzes.

3] ______ {T/F} The course outline encourages students to check the schedule page daily.

4] ______ {T/F} I realize that I should check the schedule page to see if an assignment is scheduled. (Hint TRUE)

5] Write the URL for Dr. Hicks Website on carme.

6] Write the output from printf ("%ld\n",sizeof(short int));

7] Write the output from printf ("%ld\n",sizeof(double));

8] The minimal standard for a short int is _?_ bytes; acceptable values range from _?_ to _?_.

9] The minimal standard for a long int is _?_ bytes; acceptable values range from ~_?_ billion to ~_?_ billion.

10] The minimal standard for an int is _?_ bytes; acceptable values range from _?_ to _?_.

11] The Visual Studio Net for an int is _?_ bytes; acceptable values range from ~_?_ billion to ~_?_ billion.
The Visual Studio Net short int is _?_ bytes; acceptable values range from _?_ to _?_.

The Visual Studio Net long int is _?_ bytes; acceptable values range from ~_?_ billion to ~_?_ billion.

The float is _?_ bytes and provides _?_ digits of accuracy.

The double is _?_ bytes and provides _?_ digits of accuracy.

The char is _?_ bytes.

The bool is _?_ bytes.

Write the C++ code to create an integer variable, called x, and initialize it with 5 in a single statement.

Write the C++ code to create a floating point variable, called PayRate, and initialize it with 8.25 in a single statement.

Write the C++ code to create a boolean variable, called Value, and initialize it with true in a single statement.

Write the C++ code to create an character variable, called Option, and initialize it with an A in a single statement.

Write the C++ code to create an 10 character string variable, called Name; write a single statement which will fill Name with Tom.

Hint do not use =; use strcpy!

Write the C++ code to create an 10 character string variable, called FirstName; write a single assignment statement which will fill the first character of FirstName with T.
24] Write a single assignment statement which will fill the second character of *FirstName* with o

25] Write a single assignment statement which will fill the third character of *FirstName* with m

26] Write a single assignment statement which will fill the fourth character of *FirstName* with the end of string marker *"\0"*

27] Practice with some of the formatted print statements. Assume that variable *FirstName* contains the string Tom. To the right of each of the print statements below, generate the respective output; I strongly suggest you check your answers using the computer.

```c
printf("*%s*", FirstName);   * T o m *   _ _ _ _ _
printf("*%5s*", FirstName);  _ _ _ _ _ _ _ _   _
printf("*%-5s*", FirstName);  _ _ _ _ _ _ _ _   _
printf("*%2s*", FirstName);   _ _ _ _ _ _ _ _   _
```

26] Write a statement which will display [ stdio.h ] the contents of *FirstName* in the following format:

```
First Name = Tom
```

27] Assume that variable x contains the value 123. To the right of each of the print statements below, generate the respective output; I strongly suggest you check your answers using the computer..

```c
printf("*%ld*", 54);    * 5 4 *   _ _ _ _ _
printf("*%ld*", x);    _ _ _ _ _ _ _ _   _
printf("*%5ld*", x);    _ _ _ _ _ _ _ _ _
printf("*%2ld*", x);    _ _ _ _ _ _ _ _ _
printf("*%-5ld*", x);  _ _ _ _ _ _ _ _ _
printf("*%ld*", -1*x);  _ _ _ _ _ _ _ _ _
printf("*%ld*", 6+4/2); _ _ _ _ _ _ _ _ _
```

28] Write the two statements [ stdio.h ] which will (1) prompt the user for a x and (2) enable the user to use scanf to fill the the x container defined earlier.
29] Write the two statements \[ \text{iostream.h} \] which will (1) prompt the user for a \text{x} and (2) enable the user to use \text{cin} to fill the the \text{x} container defined earlier.

30] Write the two statements \[ \text{stdio.h} \] which will (1) prompt the user for a name and (2) enable the user to use scanf to fill the the \text{Name} container defined earlier.

31] If the user enters "Tom Hicks" into the scanf above, only the first token, "Tom", is transferred into the \text{Name} container; not good. Write the two statements \[ \text{stdio.h} \] which will (1) prompt the user for a name and (2) enable the user to use \text{gets} to fill the the \text{Name} container defined earlier.

32] Write the two statements \[ \text{iostream.h} \] which will (1) prompt the user for a name and (2) enable the user to use \text{cin} to fill the the \text{Name} container defined earlier.

33] If the user enters "Tom Hicks" into the \text{cin} above, only the first token, "Tom", is transferred into the \text{Name} container; not good. Write the two statements \[ \text{iostream.h} \] which will (1) prompt the user for a name and (2) enable the user to use \text{cin} to fill the the \text{Name} container defined earlier.

34] Write the line of code that provides the compiler with the functionality to use \text{printf, scanf, puts & gets}. [HINT (include)]

35] Write the line of code that provides the compiler with the functionality to use \text{cin & cout}.

36] Write the line of code that provides the compiler with the functionality to use \text{strlen, strcmp & strcpy}.

37] Write the C++ code to create a floating point variable, called \text{PI}, and initialize it with \text{3.14} in a single statement.

38] Write the C++ code to create a floating point constant, called \text{PI}, and initialize it with \text{3.14} in a single statement. Not the same as the problem above!
39] What is the difference between a **constant** and a **variable**?

______________________________________________________________________
______________________________________________________________________

40] Write the line of code that provides the compiler with the functionality to use the functions that Dr. Hicks has provided in **Utilities.hpp/cpp**.

41] **Include Statements** are generally placed at the _?_ of the program file.

42] **Define Statements** are generally placed at immediately after the _?_ statements at the top of the file.

43] **Struct and Class Definitions** are generally placed at immediately after the _?_ statements and before the function prototypes.

44] **Function Prototypes** are generally placed at immediately after the _?_ definitions and before the functions.

45] _?_ are generally placed at the bottom of the program.

46] All local _?_ are generally defined at the top of the function.

47] Number each of the following. Place them in the order that they should normally appear in a program.

________ class and struct definitions
________ include statements
________ define statements
________ functions
________ prototypes

48] The slides describe three major reasons that programmers create functions/procedures/modules; list them.

R ______________________________  A ______________________________

E ______________________________

49] _____ {T/F} One line modules are never appropriate.

50] _____ {T/F} One line modules are generally not appropriate.

51] Write the 1 line of code for the prototype for procedure/function **DisplayGreetings**. This function is to be passed no arguments.
52) Write the complete code for function DisplayGreetings. This function is to print 25 blank lines and the message "Welcome to CSCI 2320", and 10 more blank lines.

53] Write the line of code that could be placed in function main to evoke/call DisplayGreetings.

54] Write the 1 line of code for the prototype for procedure/function Square. This function is to be passed [ by value ] an long integer argument, called No. It is to explicitly return an long integer which is the square of the passed argument.

55] Write the complete code for function Square. It is to explicitly return an long integer which is the square of the passed argument. Square(-3) should explicitly return 9.

56] Write the line of code that could be placed in function main to display [ stdio.h ] the square of -3.

57] Write the 1 line of code for the prototype for function Cube1. This function is to be passed [ by value ] an long integer argument, called No and passed an long integer pointer, called CubePtr. Function Cube1 is to have no explicit return. If the first argument passed to Cube1 is -3, then the value pointed to by CubePtr is to contain -27 upon completion of this function. Use a pointer variable!
58] Write the complete code for function **Cube1**. A long integer argument, called **No** is to be passed by value. The second argument is to be a pointer to an long integer, called **CubePtr**. Function **Cube1** is to have no explicit return (HINT: void). If the first argument passed to Cube1 is -3, then the value pointed to by **CubePtr** is to contain -27 upon completion of this function. You may not use a reference variable for this function.

59] Function main contains an integer container, called **x**. You must use function **Cube1**. Write the line of code that could be placed in function main to fill container **x** with the Cube of -3. Write the line of code to display the value now contained in **x**.

60] Write the 1 line of code for the prototype for function **Cube2**. This function is to be passed [by value] an long integer argument, called **No** and passed an long integer reference variable, called **NoCube**. Function **Cube2** is to have no explicit return. If the first argument passed to **Cube1** is -3, then the value referenced by **NoCube** is to contain -27 upon completion of this function. Use a reference variable!

61] Write the complete code for function **Cube2**. A long integer argument, called **No** is to be passed by value. The second argument is to be a reference to an long integer, called **NoCube**. Function **Cube2** is to have no explicit return (HINT: void). If the first argument passed to **Cube2** is -3, then the long integer referenced by **NoCube** is to contain -27 upon completion of this function. You must use a reference variable for this function.

62] Function main contains an integer container, called **x**. You must use function **Cube2**. Write the line of code that could be placed in function main to fill container **x** with the Cube of -3. Write the line of code to display the value now contained in **x**.

63] Write the two lines of code to fill string variable **Name** with "Nancy". (Hint-may not use =)
64] Write the line of code to fill long int variable `NoChars` with the number of characters in string variable `Name`.

65] Write the block of code to display "Matches Nancy" if and only if string variable `Name` contains the string: "Nancy".

66] There are two character string variables, called `String1` and `String2`, that are filled with uppercase/capital letters. Write the block of code to make sure that `String1` would occur earlier in an alphabetical list than `String2`. (i.e. if `String1` contains "Nancy" and `String2` contains "Andy", switch the two values.)

67] There are two character string variables, called `String1` and `String2`, that are filled with uppercase/capital letters. Write the block of code that will display the string with the most characters (longest string).

68] Write the block of code that will display the letters in `String1` - one to a line. If the string is empty, display "String Is Empty".

9] The slides describe why programming costs go down with OOP because of stacks, queues, trees, etc. Explain!
OOP is an acronym for _?_.

A function is _?_ (Evoked) by specifying its name followed by its arguments.

A function is Called (E_?_) by specifying its name followed by its arguments.

Rather than restrict the size of a function to be 50 lines of code, the slides suggest that a better guide might be "No More Than _?_ For An Experienced Computer Scientist To Understand The Complexity"

According to the slides, one of the reasons that the prototype is required is because the prototype permits _?_ of data types by the compiler.

According to the slides, one of the reasons that the prototype is required is because the prototype ensures conversion of all _?_ passed to the function to the declared argument data type when the function is called.

Write the line of code to create a int variable called No1 and initialize it with 101.

Write the line of code to display the address (in decimal) where No1 is stored. HINT: will not be 1000.

Write the line of code to display the address (in hex) where No1 is stored.

Write the line of code to display the value of the integer stored in container No1.

Write the line of code to display the size, in bytes, of container No1.

When Calling A Function By Value, the function receives a _?_ of the actual arguments passed to the function as parameters.

When Calling A Function By Value, the function can change the value of the parameter, but _?_ change the values of the actual actual arguments.

A/An _?_ is the variable which is part of the method's signature (method declaration).

A/An _?_ is an expression used when calling the method.

When calling a function by reference, the formal parameter becomes an _?_ for the actual parameter.
86]   ______________________________________________________________ When calling a function by reference, the function can change the _?_ of the actual arguments.

What To Turn In

--- No Lab Is Complete Until Both Are Complete ---

1] You sign & submit the Pledge form.
   a) Make sure that all program files have a header box with a purpose that clearly defines what you are accomplishing in this lab.
   b) Make sure that each and every program function has a well formed documentation box that clearly describes the purpose.
   c) Make sure that each and every program function header box has the appropriate Written By and Date.
   d) Review the Pledge statement
   e) Sign & Pledge
   f) Record the amount of time you think you spent on this lab
   g) Staple all pages of this lab. Fold in half length-wise (like a hot-dog). Put your name on the outside. Place it on the professor desk before the beginning of lecture on the day it is due. The penalty for late homework will not exceed 25% off per day.

2] Place all programming code associated with this program, if any, in the Professor's Code Drop Box
   a) I do not accept programs by mail; do not submit labs via email!

--- Comments ---

A] Programs that do not compile are worth little, if anything.
B] If a print statement format is off, the penalties will often be less than the 25% per day late penalty; turn in the lab. You would not be happy if you went to Best Buy and purchased a large screen TV that did everything except show the picture; you would consider it pretty worthless. Most users consider software that does not work properly pretty useless as well. If the lab is not working correctly, credit will be small (if any); you might be better to accept a 25% (1 day) late penalty and turn in the lab working correctly!
C] Start all programs early so that you can get in contact with the professor if you have problems.
D] If you are turning in this lab late, you may
   ➢ hand it to me if I am in the office
   ➢ put it in the mail box outside my office door
   ➢ slide it under the outer door to our suite (if locked)
   ➢ slide it under my office door. The sooner I get late labs, the sooner the late penalty meter quits clicking.
E] Backup your programs in at least three places. Put a copy on your Y drive. Put a copy on your flash drive. Put a copy on your personal computer. Send yourself a copy in your e-mail.