CSC 101 **Name: _____**

Lab Six(A) **Date: _____**

Objectives:

- Introduction to the debugger and debugging with gdb
- Continue refining use of conditional branching
- Introduction to Loops

Procedures:

1. Examine the following program:

```
 1 #include<iostream>
 2 using namespace std;
 3
 4 int main()
 5 {                                                x       y       z
 6     int x = 0;                                  _____   _____   _____
 7     int y = 1;                                  _____   _____   _____
 8     int z = 2;                                  _____   _____   _____
 9
10     cout << "Starting" << endl;                 _____   _____   _____
11     x = y + z;                                  _____   _____   _____
12     cout << "X is now " << x << endl;           _____   _____   _____
13     y = x * z;                                  _____   _____   _____
14     cout << "Y is now " << y << endl;           _____   _____   _____
15     z = y + x;                                  _____   _____   _____
16     cout << "Z is now " << z << endl;           _____   _____   _____
17     x = x * x;                                  _____   _____   _____
18     cout << "X is now " << x << endl;           _____   _____   _____
19
20     return 0;
21 }
```

Using the lines to the right of the code listing, trace the program, listing what the value of variables x, y, and z at each line of the program. In addition, use the box to the right list the output of this program.

Output:

2. Using a text editor, type in the above program (omitting the line numbers and tracing information). Save the program as prog7a.cpp. Compile the program, calling the output (executable) file prog7a. What command did you use to compile it?

3. Run the program. Does the output of the program match your predictions above? If not, why?

4. ***Debugging*** is the process of finding and reducing errors in our programs. Frequently, we employ the services of a program called a ***debugger***, which allows a programmer to test and debug programs by running their code inside the debugger. The debugger simulates a run of the program, allowing the programmer to stop the program at certain points and query memory locations for the current value of the variables. In order to use a debugger, an executable must contain debugging information in addition to the compiled code.

At the command prompt, issue the command:

man g++

Look through the manual page, searching for the -g compiler option. What is the function of the -g tag?

5. Compile prog7a.cpp with debugging information. Issue the command:

g++ -g -o prog7a_dbg prog7a.cpp

6. Determine the file types for each of the prog7a executables by issuing the command:

file prog7a*

Are the executable files different?

7. Determine the file sizes of the prog7a executables by issuing the command:

ls -l prog7a*

List the files and their sizes below.

8. Do the files prog7a and prog7a_dbg differ in size? If so, explain the size difference.

9. A popular debugger used with the gcc/g++ toolchain is gdb. At the command prompt, issue the command:

man gdb

After reading the manual page, describe the function and operation of the gdb debugger.

10. Start the gdb debugger by issuing the command:

gdb prog7a_dbg

What is the output of this command?

11. At the (gdb) prompt, issue the command:

help run

Describe the function and operation of the run command:

12. At the (gdb) prompt, issue the command:

run

What is the output of this command?

13. A breakpoint halts the execution of the debugged program at a certain point. Use the gdb help system to learn about breakpoints by issuing the following command at the (gdb) prompt:

help break

Describe the function and operation of the break command:

14. Set a breakpoint for the program by issuing the following command at the (gdb) prompt:
    break 10

List and explain the output of this command:

15. Run the program again by issuing the following command at the (gdb) prompt:
    run

List and explain the output of this command:

16. Learn about the print command by issuing the following command at the (gdb) prompt:
    help print

Describe the function and operation of the print command:

17. Issue the following command at the (gdb) prompt:
    print x

What is the output of this command?

18. Issue the following command at the (gdb) prompt:
    print y

What is the output of this command? Explain the output of the last two questions in reference to the current run of the program and the breakpoint set above.

19. Determine the current value of the variable z for the program. What command did you issue to determine this?

20. What is the current value of the variable z?

21. Learn about the step command by issuing the following command at the (gdb) prompt:
help step
Describe the function and operation of the step command:

22. At the (gdb) prompt, issue the command:
step
What is the output of this command?

23. At the (gdb) prompt, issue the command:
print x
What is the current value of the variable x?

24. Again issue the command:
step
Then issue the command:
print x
What is the current value of the variable x?

25. Again issue the command:
step
What is the output of this command?

26. Determine the current value of the variable y. What command did you use to do this?

27. What is the current value of the variable y?

28. Step through the next two lines of the program. What commands did you use to accomplish this?

29. Again determine the current value of the variable y and list it below. What command did you use to determine this?

30. Investigate the operation of the continue command using the gdb help system. Describe the function and operation of the continue command.

31. At the (gdb) prompt, issue the command:

    continue

What is the output of this command?

32. Exit the debugger by issuing the command:

    quit

33. Given your activities above, explain the debugging process, describing in your own terms the function and operation of each of the commands used above.

34. Examine the following program:

```
 1 #include<iostream>
 2 using namespace std;
 3
 4 int main()
 5 {
 6     int x=0;
 7     while(x<5)
 8     {
 9         cout << "X is " << x << endl;
10         x = x + 1;
11     }
12
13     return 0;
14 }
```

Trace and Output:

Use the space to the right of the program to trace it and show its output.

35. Use a text editor to type in the above program (without the line numbers). Save this program as prog7b.cpp. Compile the program (naming the executable prob7b) with debugging information. What command did you use to accomplish this?

36. Debug the prob7b program by issuing the command:
gdb prog7b

37. Run the program once in the debugger. What command did you issue to accomplish this?

38. What is the output of the program run?

39. Set a breakpoint for the prog7b program at line 9. What command did you issue to accomplish this?

40. What is the output of this command?

41. Run the program. What line is the debugger ready to execute?

42. Determine the current value of the variable x . What command did you issue to accomplish this?

43. What is the current value of the variable x?

44. At the (gdb) prompt, issue the command:
continue
What is the output of the command?

45. Again determine the current value of the variable x. What is the value?

46. Again issue the continue command, then determine the value of the variable x. What is its current value?

47. Again issue the continue command, then determine the value of the variable x. What is its current value?

48. Exit the debugger by issuing the command:
quit

49. Modify the Rock, Paper Scissors program that you wrote in the previous lab. Add appropriate code to the program so that it can be played multiple times. After playing, ask the user if he/she would like to play again. Allow the user to play as many times as they like, as long as they continue to reply in the affirmative. After adding appropriate comments, print out the modified program and hand in the source code along with this lab sheet.

50. Write a complete C++ program that implements a number guessing game. First, have the program generate a random number in the range 1 to 100. Continually prompt the user to guess the number. Have the user enter a number from the keyboard. If the guess is correct, congratulate the user, telling them how many guesses it took to reveal the number. If the guess in not correct, supply a hint, telling the user whether the number is higher or lower than the guess they supplied, then prompt the user again and read the next guess. In your program, allow the user the ability to play multiple times, using a sentinel value to let the user inform the program when they are finished. As each iteration of the game begins, display a banner, informing the user of the rules of the game.

Make sure you write a complete C++ program. You may use additional variables as necessary. Add judicious comments to document your code.

Include a header, like the one from previous labs, displaying your name, the date, and the course number.

When you have finished, print out the code and attach it to this lab sheet.