Course Name: Yu Liu

Course Number and Section: **14:332:333:01**

Experiment: [Experiment # 2 – Introduction to C Programming Language]

Lab Instructor: Ali Essam

Date Performed: 10/01/2018

Date Submitted: 10/15/2018

Submitted by: Yu Liu 173001088

Course Name: _________________

Course Number and Section: **14:332:333:01**

! Important: Please include this page in your report if the submission is a paper submission. For electronic submission (email or Sakai) please omit this page.

-------------------------For Lab Instructor Use ONLY-------------------------

GRADE: _______________

COMMENTS:

Exercise 1

1. For the code, I changed V0 from 0 to 3, V1 from -1 to 3, V2 from 0 to 1, and V3 from 0 to 3. So now it is, V0 = 3, V1 = 3, V2 = 1, and V3 = 3. WhenV0 equals 3, it will print out 3 Rus. V1 equals 3, the statement will print case 3's statement which is "Werblin Rec Center". V2 equals 1 will print the first statement " RUTGERS!" instead of "PENN STATE!". For V3 equals 3, it will print Go.

2. Macros can take arguments similar to functions. To define a macro that uses arguments, you insert parameters between the pair of parentheses in the macro definition. The number of arguments must match with the number of parameters in the macro definition. So in this code, we will have 4 macros values needed for the preprocessor macros.

3. The -o flag allows you to rename the file but that file is still linked to the original .c file that is stored in the desktop. When you call the renamed file, it will execute the original .c file.

Exercise 2

1. To set up the breakpoint in the code, type in 'break hello.c:03' after running 'gdb hello'. This will create a breakpoint at main because the ':03' means break at line 3 and the main function starts at line 3. Next type in 'display' to show the values of 'i' and '*p'. After that, type 'step' to execute the line and since it is a loop , you will repeat the count times specified. Keep typing step until it finishes all iterations and possible outcomes till you get: printf("Thanks for waddling through this program. Have a nice day."); To exit gdb mode, simply type 'quit'.

2. gdb commands I used

    a. break [file:] line

        i. break point at the line number

    b. run

        i. starts the program

    c. display

        i. show the values of the expressions

    d. step

        i. execute the line and continues if theres a loop.

3. Type 'r' then the argument to pass the command line arguments

4. Type ‘break… if’ to set a breakpoint when the conditions are true.
5. Type ‘n’ or ‘next’ to execute the next line.
6. Type ‘gdb [program]’ to start the debugging program.
7. Type ‘run’ to resume the program after break.
8. Type ‘print [/f] [expr]’ or ‘p[/f][expr]’ to see the values
9. Type ‘display [/f] expr’ to print the value of a variable after every step
10. Type ‘info variables’ to list all global and static variable names or ‘info locals’ for local variables of current stack frame.
11. Type ‘quit’ to quit/exit GDB.

Exercise 3

1. The bug was within the while loop. It successfully printed the test 1 result but it is missing its test 2 result. For this condition to work, a and b must all be true for the main function to print the second result. Therefore, by change the code to while (a & & b != NULL) it will print both tests.

Exercise 4

1. The program had no compilation error so to run cgdb, first is to crate a text file to import and export the file so the program is able to run while reading the file. Using commands such as ./int_hello <text file name, ./a.out <filename.txt, a.out <inputfile, a.out> outputfile, a.out <inputfile> outputfile. They will import or export the text file.v

Exercise 5

1.
```
int ll_has_cycle(node *head) {
 node *tortoise = head;
 node *hare = head;
 while (hare != NULL && hare -> next != NULL && hare -> next -> next != NULL)
{
    hare = hare -> next -> next;
    tortoise = tortoise -> next;
    if (tortoise == hare) {
       return 1;
    }
 }
 return 0;
}
```

For the code, I created two points, tortoise and hare. The condition of the while loop states hare will be point to 2 nodes ahead and tortoise will point to the next node. As the loop continues, the hare will be ahead of the tortoise but the hare will continue cycle through the loop until it reaches a node where both tortoise and hare share the same value. Else, if they don’t share a same node, then exit the loop.