Course Name: Comp Arch Lab 1

Course Number and Section: 14:332:333:02

Experiment: [Experiment # [2] – Intro to C programming language]

Lab Instructor: Jalal Abdulbaqi

Date Performed: 9/26/2018

Date Submitted: 10/9/2018

Submitted by: [Jahidul Islam - 171001155]

Course Name: Comp Arch Lab 1

Course Number and Section: 14:332:333:02

! 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:

Electrical and Computer Engineering Department
School of Engineering
Rutgers University, Piscataway, NJ 08854

ECE Lab Report Structure

1. Purpose / Introduction / Overview – describe the problem and provide background information
2. Approach / Method – the approach took, how problems were solved
3. Results – present your data and analysis, experimental results, etc.
4. Conclusion / Summary – what was done and how it was done

.

Introduction/Approach:

For this lab, the language of c was explored while utilizing command prompt in order to learn the basics to set the groundwork for future coding. The instructions were followed and the code was manipulated in order to acquire expected results.

Exercise 1:

Part 1:

We need to set the values to certain things displayed in the chart:

Action	Reason
V0 set to 3	Need “RU” three times in the for loop
V1 set to 3	In order to select case 3
V2 set to 1 (meaning true)	So we can print RUTGERS!
V3 set to 3	So a condition of V3==3 is met to select Go

Part 2:

For the preprocessor macros the minimum number of distinct values needed are two which are given as CONSTANT_NAME and literal_value.

Part 3:

The -o flag specifies the output executable file’s name for example “out.o”

Exercise 2:

Part 1:

In order to set the breakpoint at main use “break main”. Then to run up to it input “run”.

Part 2:

1)We can use “r” and then the arguments follow
For example:
$ gdb executablefile

(gdb) r arg1 arg2 arg3

2)We can use two different methods:
Method 1: break...if…
We can use something like:
break if variable==1

3) We use “next” to execute only the next line
4)To execute code line-by-line we use “step”

5)To continue after breaking use “continue”
6)To see the variable once we can use “print var”
7)To display value of variable after every step use “display var”
8)To list all variables and their values in the current function use “info variables”
9)In order to exit we can use “quit”

Exercise 3:

When carefully observing the function ll_equal we can see that the "if" statement checked the value of variable "a". However, it did not also check the value of variable b which is an issue. The program tried to execute "b->next" when b in our situation as we found is NULL. So we need to add an additional condition to check if b is NULL.

Exercise 4

In order to run CGDB on the executable, a simple command line is needed. One can proceed to use the command "run <name.txt". This would work because the file "name.txt" contains the user input.

Exercise 5

```
#include <stdio.h>

typedef struct node {
	int value;
	struct node *next;
} node;

int ll_has_cycle(node *head) {
	/* your code here */
	node *tortoise = head, *hare = head; //In this part we are essentially setting up variables
	do
	{
		//Now we proceed to check and make sure  the pointer’s values aren't NULL
		//because that would cause errors when checking its value
		if(hare == NULL || hare->next == NULL || hare->next->next == NULL)
			return 0;
		//This next part will make hare advance by two nodes
		hare = hare->next->next;
		//Afterwards the next line makes the tortoise advance by one node and a NULL
		//pointer is not necessary because it only moves after hare is checked
		tortoise = tortoise->next;
	//It will continue to do it until the hare finally reaches the tortoise
	}while (hare != tortoise);
	return 1;
}
```

```
void test_ll_has_cycle(void) {
	int i;
	node nodes[25]; //enough to run our tests
	for(i=0; i < sizeof(nodes)/sizeof(node); i++) {
		nodes[i].next = 0;
		nodes[i].value = 0;
	}
	nodes[0].next = &nodes[1];
	nodes[1].next = &nodes[2];
	nodes[2].next = &nodes[3];
	printf("Checking first list for cycles. There should be none, ll_has_cycle says it has %s
cycle\n", ll_has_cycle(&nodes[0])?"a":"no");

	nodes[4].next = &nodes[5];
	nodes[5].next = &nodes[6];
	nodes[6].next = &nodes[7];
	nodes[7].next = &nodes[8];
	nodes[8].next = &nodes[9];
	nodes[9].next = &nodes[10];
	nodes[10].next = &nodes[4];
	printf("Checking second list for cycles. There should be a cycle, ll_has_cycle says it has
%s cycle\n", ll_has_cycle(&nodes[4])?"a":"no");

	nodes[11].next = &nodes[12];
	nodes[12].next = &nodes[13];
	nodes[13].next = &nodes[14];
	nodes[14].next = &nodes[15];
	nodes[15].next = &nodes[16];
	nodes[16].next = &nodes[17];
	nodes[17].next = &nodes[14];
	printf("Checking third list for cycles. There should be a cycle, ll_has_cycle says it has %s
cycle\n", ll_has_cycle(&nodes[11])?"a":"no");

	nodes[18].next = &nodes[18];
	printf("Checking fourth list for cycles. There should be a cycle, ll_has_cycle says it has
%s cycle\n", ll_has_cycle(&nodes[18])?"a":"no");

	nodes[19].next = &nodes[20];
	nodes[20].next = &nodes[21];
	nodes[21].next = &nodes[22];
	nodes[22].next = &nodes[23];
	printf("Checking fifth list for cycles. There should be none, ll_has_cycle says it has %s
cycle\n", ll_has_cycle(&nodes[19])?"a":"no");
```

```
        printf("Checking length-zero list for cycles. There should be none, ll_has_cycle says it
has %s cycle\n", ll_has_cycle(NULL)?"a":"no");
}

int main(void) {
  test_ll_has_cycle();
  return 0;
}
```

Conclusion:

Overall, the lab was a basic introduction to the language of c and completing it provided the general rules and syntax that we will be using for the semester ahead. The code itself was almost all provided so manipulation in order to understand was a fun yet informative task. Success was achieved as results displayed were the ones expected.