Exercise 1

```
#include <stdio.h>

/* Only change any of these 4 values */
#define V0 3
#define V1 3
#define V2 1
#define V3 3

int main(void) {
	int a;
	char *s;

	/* This is a print statement. Notice the little 'f' at the end!
	It might be worthwhile to look up how printf works for your future
	debugging needs... */
	printf("Simple C program:\n====================\n");

	/* for loop */
	for(a=0; a<V0; a++) {
		printf("RU ");
	}
	printf("\n");

	/* switch statement */
	switch(V1) {
		case 0:		printf("Werblin Rec Center\n");
		case 1:		printf("Busch Campus Center\n");	break;
		case 2:		printf("Livingston \n");
		case 3:		printf("Werblin Rec Center\n");			break;
		case 4:		printf("CoRE\n");	break;
		case 5:		printf("ECE\n");
		default:	printf("Hill Center\n");
	}

	/* ternary operator */
	s = (V3==3) ? "Go" : "Boo";

	/* if statement */
	if(V2) {
		printf("\n%s RUTGERS!\n",s);
	} else  {
```

```
                printf("\n%s PENN STATE!\n",s);
        }

        return 0;
}
```

```
yl1025@ece16:~$ cd ~/Desktop/Lab2
yl1025@ece16:~/Desktop/Lab2$ gcc -o simple simple.c
yl1025@ece16:~/Desktop/Lab2$ ./simple
Simple C program:
====================
RU RU RU
Werblin Rec Center

Go RUTGERS!
yl1025@ece16:~/Desktop/Lab2$
```

```
`/#include <stdio.h>

/* Only change any of these 4 values */
#define V0 3
#define V1 3
#define V2 1
#define V3 3

int main(void) {
	int a;
	char *s;

	/* This is a print statement. Notice the little 'f' at the end!
	It might be worthwhile to look up how printf works for your future
	debugging needs... */
	printf("Simple C program:\n====================\n");

	/* for loop */
	for(a=0; a<V0; a++) {
		printf("RU ");
	}
	printf("\n");

	/* switch statement */
	switch(V1) {
		case 0:		printf("Werblin Rec Center\n");
		case 1:		printf("Busch Campus Center\n");	break;
		case 2:		printf("Livingston \n");
		case 3:		printf("Werblin Rec Center\n");			break;
		case 4:		printf("CoRE\n");	break;
		case 5:		printf("ECE\n");
		default:	printf("Hill Center\n");
	}

	/* ternary operator */
	s = (V3==3) ? "Go" : "Boo";

	/* if statement */
	if(V2) {
		printf("\n%s RUTGERS!\n",s);
	} else  {
```

```
                printf("\n%s PENN STATE!\n",s);
        }

        return 0;
}
```

```
yl1025@ece16:~/Desktop/Lab2$ gcc -g -o hello hello.c
yl1025@ece16:~/Desktop/Lab2$ gdb hello
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello...done.
(gdb) break hello.c:03
Breakpoint 1 at 0x4005a5: file hello.c, line 3.
(gdb) run
Starting program: /home/user/Desktop/Lab2/hello

Breakpoint 1, main (argc=1, argv=0x7fffffffe3f8) at hello.c:4
4       {
(gdb) display i
1: i = 0
(gdb) display *p
2: *p = 1
(gdb) step
5           int i, *p, count = 0;
1: i = 0
2: *p = 1
(gdb) step
6           p = &count;
1: i = 0
2: *p = 1
(gdb) step
```

```
8           for (i = 0; i < 10; i++) {
1: i = 0
2: *p = 0
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 0
2: *p = 0
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 0
2: *p = 1
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 1
2: *p = 1
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 1
2: *p = 2
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 2
2: *p = 2
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 2
2: *p = 3
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 3
2: *p = 3
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 3
2: *p = 4
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 4
```

```
2: *p = 4
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 4
2: *p = 5
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 5
2: *p = 5
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 5
2: *p = 6
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 6
2: *p = 6
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 6
2: *p = 7
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 7
2: *p = 7
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 7
2: *p = 8
(gdb) step
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 8
2: *p = 8
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 8
2: *p = 9
(gdb) step
```

```
9                   (*p)++; //Do you understand this line of code and all the other permutations of the
operators? ;)
1: i = 9
2: *p = 9
(gdb) step
8           for (i = 0; i < 10; i++) {
1: i = 9
2: *p = 10
(gdb) step
12          printf("Thanks for waddling through this program. Have a nice day.");
1: i = 10
2: *p = 10
(gdb)
```

Exercise 3

```
#include <stdio.h>

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

/* FIXME: this function is buggy. */
int ll_equal(const node* a, const node* b) {
	while (a && b) {
		if (a->val != b->val)
			return 0;
		a = a->next;
		b = b->next;
	}
	/* lists are equal if a and b are both null */
	return a == b;
}

/* The main function exists just to test ll_equal.
   There are two tests. The second one is by default
   buggy. Please find the error and fix it! */
int main(int argc, char** argv) {
	int i;
	node nodes[10];

	for (i=0; i<10; i++) {
		nodes[i].val = 0;
		nodes[i].next = NULL;
	}

	nodes[0].next = &nodes[1];
	nodes[1].next = &nodes[2];
	nodes[2].next = &nodes[3];

	printf("equal test 1 result = %d\n", ll_equal(&nodes[0], &nodes[0]));
	printf("equal test 2 result = %d\n", ll_equal(&nodes[0], &nodes[2]));

	return 0;
}
```

```
yl1025@ece16:~/Desktop/Lab2$ gcc -g -o ll_equal ll_equal.c
yl1025@ece16:~/Desktop/Lab2$ ./ll_equal
equal test 1 result = 1
equal test 2 result = 0
yl1025@ece16:~/Desktop/Lab2$
```