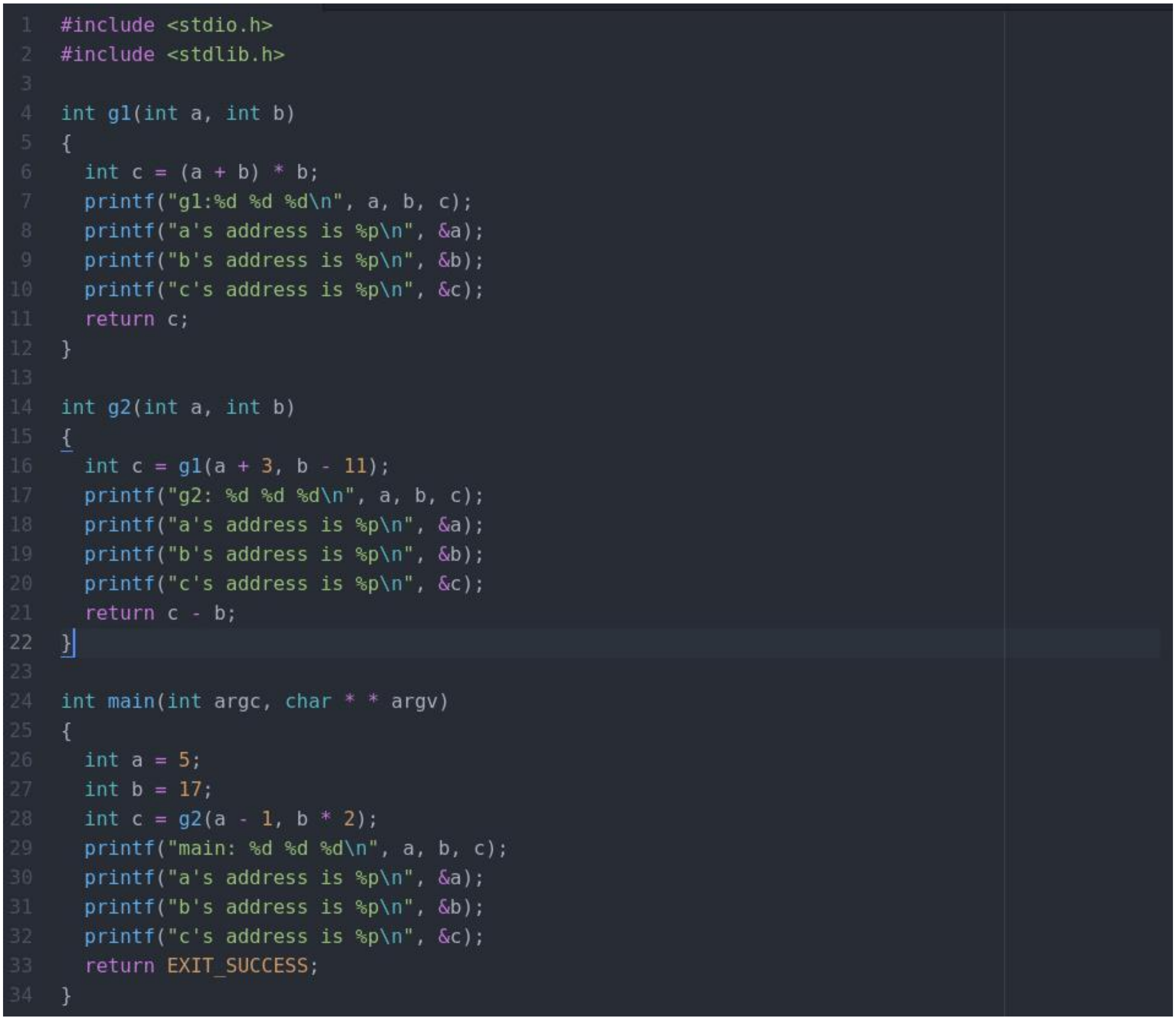
CS 2263 - FR01A

Lab 2

By Ngoc Phuong Anh Nguyen - 3712361

September 2021

Exercise One:



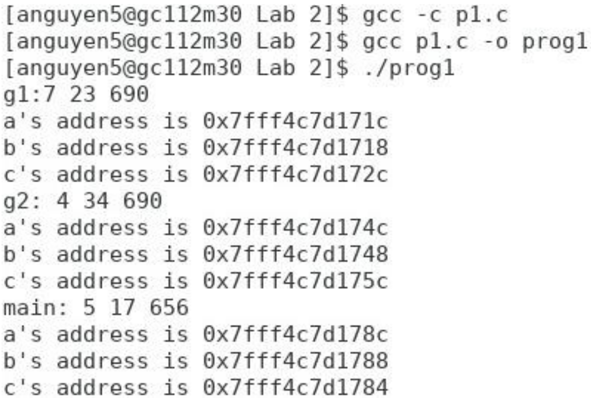
```c
#include <stdio.h>
#include <stdlib.h>

int g1(int a, int b)
{
  int c = (a + b) * b;
  printf("g1:%d %d %d\n", a, b, c);
  printf("a's address is %p\n", &a);
  printf("b's address is %p\n", &b);
  printf("c's address is %p\n", &c);
  return c;
}

int g2(int a, int b)
{
  int c = g1(a + 3, b - 11);
  printf("g2: %d %d %d\n", a, b, c);
  printf("a's address is %p\n", &a);
  printf("b's address is %p\n", &b);
  printf("c's address is %p\n", &c);
  return c - b;
}

int main(int argc, char * * argv)
{
  int a = 5;
  int b = 17;
  int c = g2(a - 1, b * 2);
  printf("main: %d %d %d\n", a, b, c);
  printf("a's address is %p\n", &a);
  printf("b's address is %p\n", &b);
  printf("c's address is %p\n", &c);
  return EXIT_SUCCESS;
}
```

Figure 1: Modified Source Code of Exercise One



```
[anguyen5@gc112m30 Lab 2]$ gcc -c p1.c
[anguyen5@gc112m30 Lab 2]$ gcc p1.c -o prog1
[anguyen5@gc112m30 Lab 2]$ ./prog1
g1:7 23 690
a's address is 0x7fff4c7d171c
b's address is 0x7fff4c7d1718
c's address is 0x7fff4c7d172c
g2: 4 34 690
a's address is 0x7fff4c7d174c
b's address is 0x7fff4c7d1748
c's address is 0x7fff4c7d175c
main: 5 17 656
a's address is 0x7fff4c7d178c
b's address is 0x7fff4c7d1788
c's address is 0x7fff4c7d1784
```

Figure 2: Output Result of Exercise One

Question 1: Are the values of the variables printed from your program the same as obtained by your colleagues? Why?

All the values of the variables printed from my program are the same as my colleagues' programs. It is because we put the same value to each variable. (a = 5, b = 17, c = g2(a - 1, b * 2))

Question 2: Are the addresses printed from your program the same as obtained by your colleagues? Why?

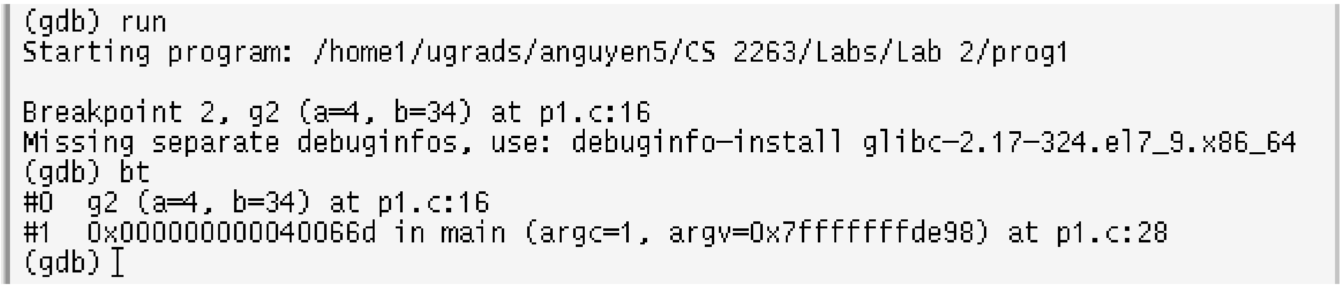
The addresses printed from my program are different from obtained by my colleagues. The reason is that the location of the memory available for C program will begin in different places depending on what is running on the computer as well as the RAM capacity.

Question 3: Are the addresses printed for the variables in the function g1 bigger or smaller than the addresses printed from the function g2? Why?

The addresses printed for the variables in the function g1 are smaller than the addresses printed for the variables in the function g2. The reason is because g1 was called after g2, so it would be put on the top of the stack.

Exercise Two:

Question 1: Submit the debugger screenshot showing the backtrace after reaching the breakpoint g2. How many frames are shown in the trace?

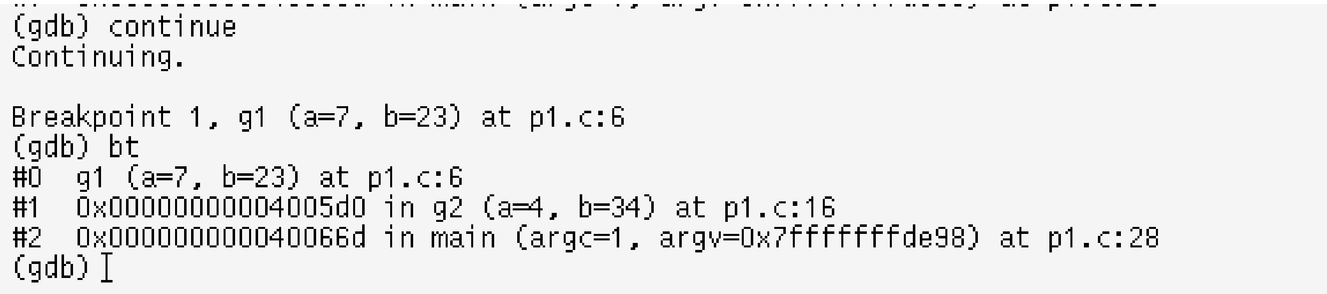


```
(gdb) run
Starting program: /home1/ugrads/anguyen5/CS 2263/Labs/Lab 2/prog1

Breakpoint 2, g2 (a=4, b=34) at p1.c:16
Missing separate debuginfos, use: debuginfo-install glibc-2.17-324.el7_9.x86_64
(gdb) bt
#0  g2 (a=4, b=34) at p1.c:16
#1  0x000000000040066d in main (argc=1, argv=0x7fffffffde98) at p1.c:28
(gdb) 
```

There are 2 frames in the trace.

Question 2: Submit the debugger screenshot showing the backtrace after reaching the breakpoint g1. How many frames are shown in the trace?



```
(gdb) continue
Continuing.

Breakpoint 1, g1 (a=7, b=23) at p1.c:6
(gdb) bt
#0  g1 (a=7, b=23) at p1.c:6
#1  0x00000000004005d0 in g2 (a=4, b=34) at p1.c:16
#2  0x000000000040066d in main (argc=1, argv=0x7fffffffde98) at p1.c:28
(gdb) 
```

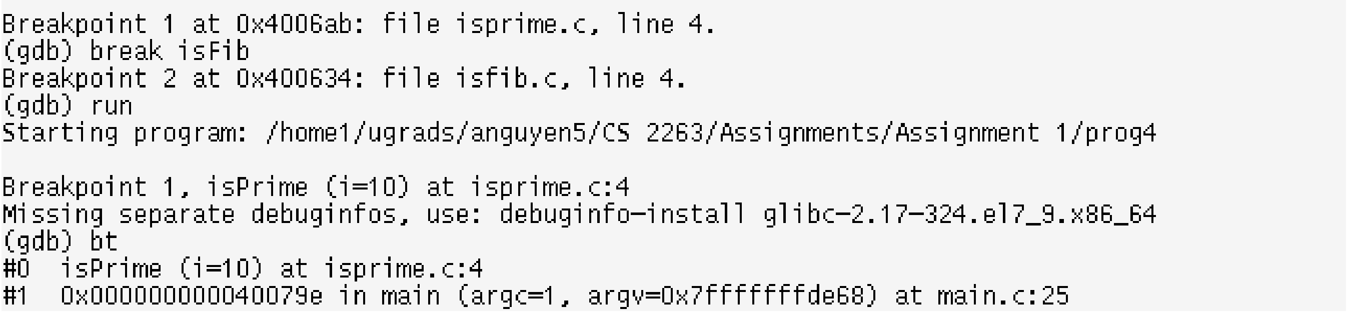
There are 3 frames in the trace.

Question 3: Are the stack addresses listed in backtrace related to the addresses of the variables a, b and c printed by the functions g1 and g2? Explain.

The addresses shown in DDD are not the actual memory addresses, they are offsets from some starting address that DDD picked

Exercise Three:

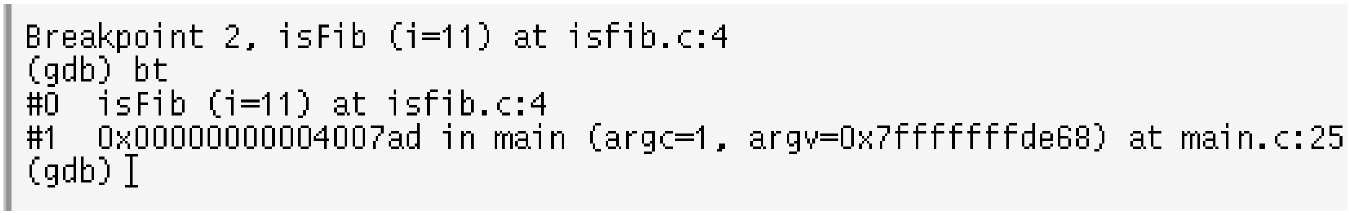
Question: Run your program from your assignment 1 for finding the Fibonacci primes in the debugger. Set the breakpoints to the functions isFib and isPrime



```
Breakpoint 1 at 0x4006ab: file isprime.c, line 4.
(gdb) break isFib
Breakpoint 2 at 0x400634: file isfib.c, line 4.
(gdb) run
Starting program: /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 1/prog4

Breakpoint 1, isPrime (i=10) at isprime.c:4
Missing separate debuginfos, use: debuginfo-install glibc-2.17-324.el7_9.x86_64
(gdb) bt
#0  isPrime (i=10) at isprime.c:4
#1  0x000000000040079e in main (argc=1, argv=0x7fffffffde68) at main.c:25
```

Figure 3: Debugger screenshot showing the backtrace after reaching the breakpoint isPrime



```
Breakpoint 2, isFib (i=11) at isfib.c:4
(gdb) bt
#0  isFib (i=11) at isfib.c:4
#1  0x00000000004007ad in main (argc=1, argv=0x7fffffffde68) at main.c:25
(gdb) 
```

Figure 4: Debugger screenshot showing the backtrace after reaching the breakpoint isFib