Daniyal Khan

3765942

CS-2263

Lab 2

Exercise 1:

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

int g1(int a, int b)
{
    int c = (a + b) * b;
    putchar('\n');
    printf("Address of a in g1: %p\n", &a);
    printf("Address of b in g1: %p\n", &b);
    printf("Address of c in g1: %p\n", &c);
    printf("g1: %d %d %d\n", a, b, c);
    return c;
}

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

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

Output of the program:

```
Address of a in g1: 0x7fffffffde2c
Address of b in g1: 0x7fffffffde28
Address of c in g1: 0x7fffffffde3c
g1: 7 23 690

Address of a in g2: 0x7fffffffde5c
Address of b in g2: 0x7fffffffde58
Address of c in g2: 0x7fffffffde6c
g2: 4 34 690

Address of a in main: 0x7fffffffde9c
Address of b in main: 0x7fffffffde98
Address of c in main: 0x7fffffffde94
main: 5 17 656
```

The values of the variables printed from my program is the same as my colleague because we are using the same values for a, b and c. And they don’t change.

The addresses of the variables printed from my program are different than my colleagues because every machine/computer stores them at a different place in memory.

The address printed for the variables in the function g1 is smaller than g2 because g1 is at the top of stack. In C language the memory address is allocated in descending order.

Exercise 2:

Backtrace after reaching the breakpoint g2:

```
Starting program: /home1/ugrads/v4rvh/CS2263/Labs/Lab2/p1
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

Breakpoint 2, g2 (a=4, b=34) at Exercise1.c:17
17          int c = g1(a + 3, b - 11);
(gdb) bt
#0  g2 (a=4, b=34) at Exercise1.c:17
#1  0x0000000000401288 in main (argc=1, argv=0x7fffffffdfb8) at Exercise1.c:30
```

There are 2 frames shown in the trace.

Backtrace after reaching the breakpoint g1:

```
(gdb) continue
Continuing.

Breakpoint 1, g1 (a=7, b=23) at Exercise1.c:6
6           int c = (a + b) * b;
(gdb) bt
#0  g1 (a=7, b=23) at Exercise1.c:6
#1  0x00000000004011e3 in g2 (a=4, b=34) at Exercise1.c:17
#2  0x0000000000401288 in main (argc=1, argv=0x7fffffffdfb8) at Exercise1.c:30
```

There are 3 frames shown in the trace.

At breakpoint g2:

Frame 0:

```
(gdb) frame 0
#0  g2 (a=4, b=34) at Exercise1.c:17
17          int c = g1(a + 3, b - 11);
(gdb) info frame 0
Stack frame at 0x7fffffffde80:
 rip = 0x4011ce in g2 (Exercise1.c:17); saved rip = 0x401288
 called by frame at 0x7fffffffdeb0
 source language c.
 Arglist at 0x7fffffffde70, args: a=4, b=34
 Locals at 0x7fffffffde70, Previous frame's sp is 0x7fffffffde80
 Saved registers:
  rbp at 0x7fffffffde70, rip at 0x7fffffffde78
```

Frame 1:

```
(gdb) frame 1
#1  0x0000000000401288 in main (argc=1, argv=0x7fffffffdfb8) at Exercise1.c:30
30          int c = g2(a - 1, b * 2);
(gdb) info frame 1
Stack frame at 0x7fffffffdeb0:
 rip = 0x401288 in main (Exercise1.c:30); saved rip = 0x7ffff7c29590
 caller of frame at 0x7fffffffde80
 source language c.
 Arglist at 0x7fffffffdea0, args: argc=1, argv=0x7fffffffdfb8
 Locals at 0x7fffffffdea0, Previous frame's sp is 0x7fffffffdeb0
 Saved registers:
  rbp at 0x7fffffffdea0, rip at 0x7fffffffdea8
(gdb)
```

At breakpoint g1:

Frame 0:

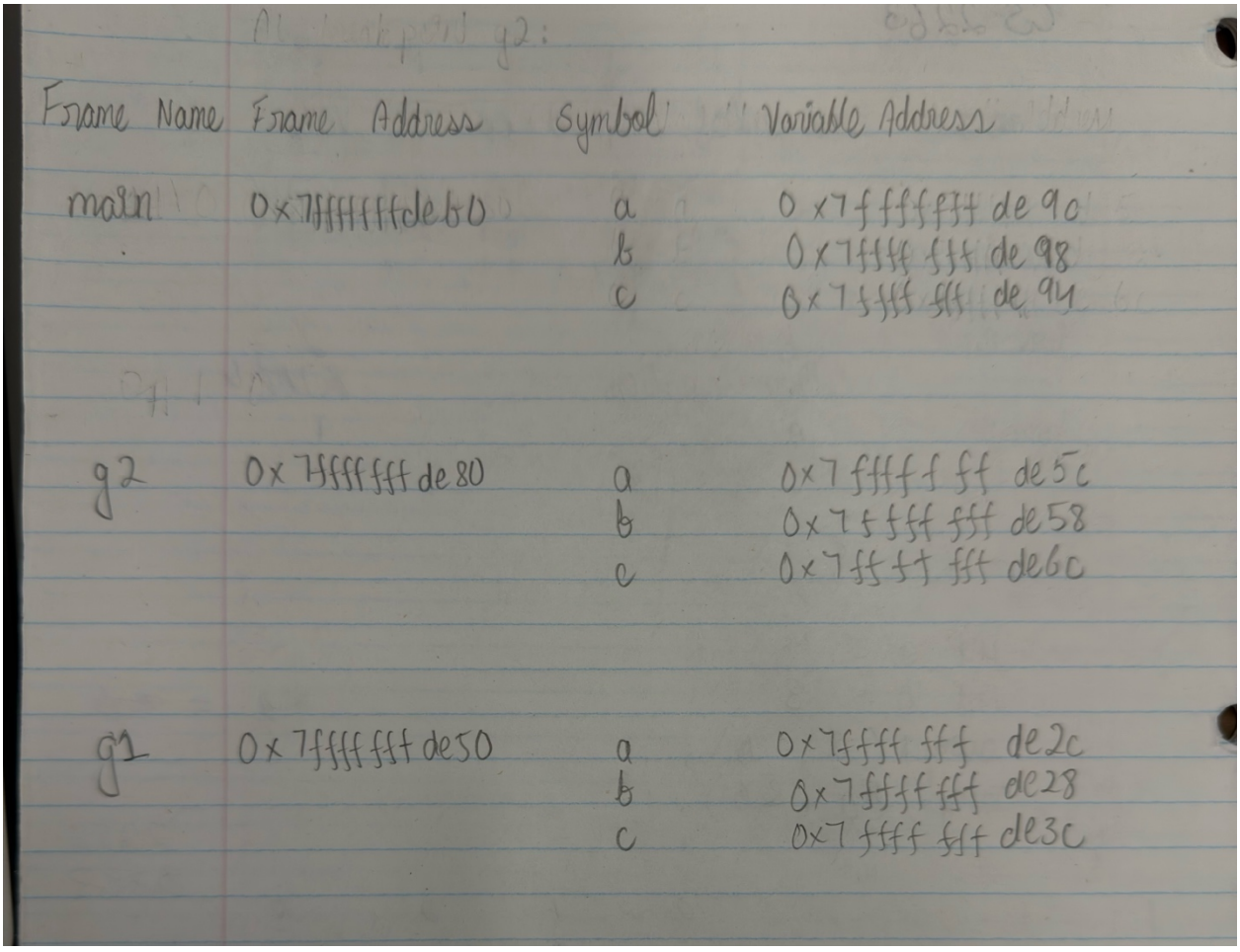
```
(gdb) info frame 0
Stack frame at 0x7fffffffde50:
 rip = 0x401144 in g1 (Exercise1.c:6); saved rip = 0x4011e3
 called by frame at 0x7fffffffde80
 source language c.
 Arglist at 0x7fffffffde40, args: a=7, b=23
 Locals at 0x7fffffffde40, Previous frame's sp is 0x7fffffffde50
 Saved registers:
  rbp at 0x7fffffffde40, rip at 0x7fffffffde48
```

Frame 1:

```
(gdb) info frame 1
Stack frame at 0x7fffffffde80:
 rip = 0x4011e3 in g2 (Exercise1.c:17); saved rip = 0x401288
 called by frame at 0x7fffffffdeb0, caller of frame at 0x7fffffffde50
 source language c.
 Arglist at 0x7fffffffde70, args: a=4, b=34
 Locals at 0x7fffffffde70, Previous frame's sp is 0x7fffffffde80
 Saved registers:
  rbp at 0x7fffffffde70, rip at 0x7fffffffde78
(gdb)
```

Frame 2:

```
(gdb) info frame 2
Stack frame at 0x7fffffffdeb0:
 rip = 0x401288 in main (Exercise1.c:30); saved rip = 0x7ffff7c29590
 caller of frame at 0x7fffffffde80
 source language c.
 Arglist at 0x7fffffffdea0, args: argc=1, argv=0x7fffffffdfb8
 Locals at 0x7fffffffdea0, Previous frame's sp is 0x7fffffffdeb0
 Saved registers:
  rbp at 0x7fffffffdea0, rip at 0x7fffffffdea8
```



NOTE: My memory map has main on the top because as I was going through the program I just wrote main first because it gets called first but I do understand the fact that main would be at the bottom of the stack.

Stack addresses are related to the addresses of the variables as we can see that for the g2 stack the address is 0x7fffffffde80 and the variables lie between that and the stack address of the next stack which is g1 is this case 0x7fffffffde50.
The address of a b and c lie between g2's address and g1's address.

Exercise 3:

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

void calcFib(int n);

int main() {
    calcFib(10);
    return EXIT_SUCCESS;
}

void calcFib(int n) {
    int a = 0, b = 1, temp;
    if (n <= 0) {
        return;
    }

    printf("%d ", a);
    for (int i = 1; i < n; i++) {
        printf("%d ", b);
        temp = a + b;
        a = b;
        b = temp;
    }
}
```

Debugger screenshot showing the backtrace after reaching breakpoint at calcFib: