Prog1:

```
Print Results:
Address 1 = 0x7ffe87a17930    Address 2 = 0x55c13b93a260
Address 1 = 0x7ffe879e7380    Address 2 = 0x55c13b93a870
Address 1 = 0x7ffe879b6dd0    Address 2 = 0x55c13b93aa70
Address 1 = 0x7ffe87986820    Address 2 = 0x55c13b93ac70
Address 1 = 0x7ffe87956270    Address 2 = 0x55c13b93ae70
Address 1 = 0x7ffe87925cc0    Address 2 = 0x55c13b93b070
Address 1 = 0x7ffe878f5710    Address 2 = 0x55c13b93b270
Address 1 = 0x7ffe878c5160    Address 2 = 0x55c13b93b470
Address 1 = 0x7ffe87894bb0    Address 2 = 0x55c13b93b670
Address 1 = 0x7ffe87864600    Address 2 = 0x55c13b93b870
```

a) Address 1s are the local addresses, Address 2s are dynamically allocated addresses

Address 1s grows from higher address to lower address, which is in the stack.
Address 2s grows from lower address to higher address, which is in the heap.

Stack expands from higher to lower address.
Heap expands from lower to higher address.

```
7fff74bca000-7fff74ddf000 rw-p 00000000 00:00 0
[stack]
Size:               2132 kB
```

b) the stack size is 2132 kB during waiting for input

```
55d721af4000-55d721b15000 rw-p 00000000 00:00 0
[heap]
Size:                132 kB
```

c) the heap size is 132 kB during waiting for input

d) The address limit for stack:

```
7f4671bbc000-7f4671da3000 r-xp 00000000 08:01 778278
/lib/x86_64-linux-gnu/libc-2.27.so
7f4671da3000-7f4671fa3000 ---p 001e7000 08:01 778278
/lib/x86_64-linux-gnu/libc-2.27.so
7f4671fa3000-7f4671fa7000 r--p 001e7000 08:01 778278
/lib/x86_64-linux-gnu/libc-2.27.so
```

```
7f4671fa7000-7f4671fa9000 rw-p 001eb000 08:01 778278
/lib/x86_64-linux-gnu/libc-2.27.so
7f4671fa9000-7f4671fad000 rw-p 00000000 00:00 0
7f4671fad000-7f4671fd6000 r-xp 00000000 08:01 778259
/lib/x86_64-linux-gnu/ld-2.27.so
7f46721d4000-7f46721d6000 rw-p 00000000 00:00 0
7f46721d6000-7f46721d7000 r--p 00029000 08:01 778259
/lib/x86_64-linux-gnu/ld-2.27.so
7f46721d7000-7f46721d8000 rw-p 0002a000 08:01 778259
/lib/x86_64-linux-gnu/ld-2.27.so
7f46721d8000-7f46721d9000 rw-p 00000000 00:00 0
7ffe419e8000-7ffe41bff000 rw-p 00000000 00:00 0
[stack]
```
The address limit for heap:

```
55b669f1d000-55b669f3e000 rw-p 00000000 00:00 0
[heap]
```

e)

```
e5-cse-204-10.cse.psu.edu 26$ strace prog1
execve("./prog1", ["prog1"], 0x7fff10061c80 /* 50 vars */) = 0
```
//This executes the prog1
```
brk(NULL)   = 0xa8f000
```
//This means no break points

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f10606e3000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)//Attempt to access a file and check existence of it
open("/usr/local/cuda/lib64/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (No such file or directory)//Opens a file
stat("/usr/local/cuda/lib64/tls/x86_64", 0x7fff55a71f50) = -1 ENOENT
(No such file or directory)
open("/usr/local/cuda/lib64/tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
stat("/usr/local/cuda/lib64/tls", 0x7fff55a71f50) = -1 ENOENT (No such
file or directory)//
open("/usr/local/cuda/lib64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) =
-1 ENOENT (No such file or directory)
stat("/usr/local/cuda/lib64/x86_64", 0x7fff55a71f50) = -1 ENOENT (No
such file or directory)
open("/usr/local/cuda/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
stat("/usr/local/cuda/lib64", {st_mode=S_IFDIR|0755,
st_size=8192, ...}) = 0
```

```
open("tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)
open("tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
open("x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)
open("libc.so.6", O_RDONLY|O_CLOEXEC)   = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=204488, ...}) = 0//
mmap(NULL, 204488, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f10606b1000//
close(3)                                = 0
open("/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`&\2\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2156272, ...}) = 0
mmap(NULL, 3985920, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f10600f5000
mprotect(0x7f10602b9000, 2093056, PROT_NONE) = 0
mmap(0x7f10604b8000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1c3000) = 0x7f10604b8000
mmap(0x7f10604be000, 16896, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f10604be000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f10606b0000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f10606ae000
arch_prctl(ARCH_SET_FS, 0x7f10606ae740) = 0
mprotect(0x7f10604b8000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7f10606e4000, 4096, PROT_READ) = 0
munmap(0x7f10606b1000, 204488)          = 0
brk(NULL)                               = 0xa8f000
brk(0xab0000)                           = 0xab0000
brk(NULL)                               = 0xab0000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 37), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f10606e2000
write(1, "Address 1 = 0x7fff55a43210    Ad"..., 51Address 1 =
0x7fff55a43210    Address 2 = 0xa8f010
) = 51
write(1, "Address 1 = 0x7fff55a12c70    Ad"..., 51Address 1 =
0x7fff55a12c70    Address 2 = 0xa8f210
) = 51
write(1, "Address 1 = 0x7fff559e26d0    Ad"..., 51Address 1 =
0x7fff559e26d0    Address 2 = 0xa8f410
) = 51
```

```
write(1, "Address 1 = 0x7fff559b2130    Ad"..., 51Address 1 =
0x7fff559b2130    Address 2 = 0xa8f610
) = 51
write(1, "Address 1 = 0x7fff55981b90    Ad"..., 51Address 1 =
0x7fff55981b90    Address 2 = 0xa8f810
) = 51
write(1, "Address 1 = 0x7fff559515f0    Ad"..., 51Address 1 =
0x7fff559515f0    Address 2 = 0xa8fa10
) = 51
write(1, "Address 1 = 0x7fff55921050    Ad"..., 51Address 1 =
0x7fff55921050    Address 2 = 0xa8fc10
) = 51
write(1, "Address 1 = 0x7fff558f0ab0    Ad"..., 51Address 1 =
0x7fff558f0ab0    Address 2 = 0xa8fe10
) = 51
write(1, "Address 1 = 0x7fff558c0510    Ad"..., 51Address 1 =
0x7fff558c0510    Address 2 = 0xa90010
) = 51
write(1, "Address 1 = 0x7fff5588ff70    Ad"..., 51Address 1 =
0x7fff5588ff70    Address 2 = 0xa90210
) = 51
write(1, "Enter any key to exit\n", 22Enter any key to exit
) = 22
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 37), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f10606e1000
read(0,
"\n", 1024)                     = 1
exit_group(0)                           = ?
+++ exited with 0 +++
e5-cse-204-10.cse.psu.edu 27$
```

Prog2:

a):

32:

```
Entering Function

Stack Address  = 0xffc3bc3c    Heap Address = 0x8f26008

Stack Address  = 0xffb16c8c    Heap Address = 0x8f2d540

Stack Address  = 0xff9f1cdc    Heap Address = 0x8f34a78

Stack Address  = 0xff8ccd2c    Heap Address = 0x8f3bfb0

Stack Address  = 0xff7a7d7c    Heap Address = 0x8f434e8

Stack Address  = 0xff682dcc    Heap Address = 0x8f4aa20

Segmentation fault (core dumped)
```

64:

```
Entering Function

Stack Address  = 0x7ffc39092150    Heap Address = 0x1a79010

Stack Address  = 0x7ffc38f6d1a0    Heap Address = 0x1a80550

Stack Address  = 0x7ffc38e481f0    Heap Address = 0x1a87a90

Stack Address  = 0x7ffc38d23240    Heap Address = 0x1a8efd0

Stack Address  = 0x7ffc38bfe290    Heap Address = 0x1a96510

Stack Address  = 0x7ffc38ad92e0    Heap Address = 0x1a9da50

Segmentation fault (core dumped)
```

The stack address and heap address are within different range
The stack address for 64 is 8bytes while for 32 is 4 bytes

i. size of compiled code:

8.4K for prog2_32
9.8K for prog2_64

```
-rwx------ 1 wml5141 e5-cse-ucse 8.4K Feb 16 16:32 prog2_32
-rwx------ 1 wml5141 e5-cse-ucse 9.8K Feb 16 16:32 prog2_64
```

ii. By using the top:
    prog2_32: 9352 kB
    prog2_32: 11428 kB
iii:
Shared Lib for 32: 232kB
Shared Lib for 64: 272kB

b) For prog_2_32

Program received signal SIGSEGV, Segmentation fault.
allocate (count=4) at prog2.c:11
11   c = malloc (30000);

For prog2_64

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400618 in allocate (
    count=<error reading variable: Cannot access memory at address 0x7fffff7fae5c>) at prog2.c:5
which is line 4 that allocate() can't read "count"

c):
For 32:
Because the segmentation fault happens at the mallc(), this should be related to heap:
Limit for stack in prog2_32:
ff91e000-ffffe000 rw-p 00000000 00:00 0
[stack]

Limit for heap in pro2_32:
0804b000-08097000 rw-p 00000000 00:00 0
[heap]

Here is the disassemble of allocate when the bug occurr, which is the location for malloc (30000)

```
   0x080484a6 <+9>:  cmpl   $0x0,0x8(%ebp)
   0x080484aa <+13>: js     0x80484e6 <allocate+73>
=> 0x080484ac <+15>: movl   $0x7530,(%esp)
   0x080484b3 <+22>: call   0x8048360 <malloc@plt>
   0x080484b8 <+27>: mov    %eax,-0xc(%ebp)
```

we also find something wired in print results:

Stack Address = 0xffed7e4c   Heap Address = 0x804b008

Stack Address = 0xffdb2e9c   Heap Address = 0x8052540

Stack Address = 0xffc8deec   Heap Address = 0x8059a78

Stack Address = 0xffb68f3c   Heap Address = 0x8060fb0

Stack Address = 0xffa43f8c   Heap Address = 0x80684e8

Stack Address = 0xff91efdc   Heap Address = 0x806fa20

The lower bound for stack is 0xFF910000

The next x is gonna be 0xFF7FA02C

as we know address of c is just near the address of x
The core dump was due to assigning the heap address to c(When c is in a illegal mem space) while this c's address is illegal to be modified. This should be a stack over flow and accessing the illegal mem.

The cause of this was due to   int x[3000000];
it costs about 1200kB each time while the stack only has about 7200kB, so it eventually busts the stack at the seventh iteration

For 64:

Stack Address = 0x7fffffed8c80   Heap Address = 0x602010

Stack Address = 0x7fffffdb3cd0   Heap Address = 0x609550

Stack Address = 0x7fffffc8ed20   Heap Address = 0x610a90

Stack Address = 0x7fffffb69d70   Heap Address = 0x617fd0

Stack Address = 0x7fffffa44dc0   Heap Address = 0x61f510

Stack Address = 0x7fffff91fe10   Heap Address = 0x626a50

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400618 in allocate (
    count=<error reading variable: Cannot access memory at address 0x7fffff7fae5c>) at prog2.c:5
5   {

The stack frame:
7fffff91f000-7fffffffff000 rw-p 00000000 00:00 0
[stack]

then that is the same issue: 0x7fffff7fae5c is out of the bound, stack over flow or accessing the illegal mem.

The same issue as prog2_32.

The cause is “int x[3000000];” for the same reason.

d)

The difference between addresses between each iteration by the results above is about 1200kB.In order to accomplish this program, we need at least 12000kB For Stack and 300kB for heap.

e)Generally, in a stack frame, there should be local variables for the function, the return address, stack pointer and frame pointer.

x and c are local variables;

x has about 1200 kB size as an array;(in both 32/64 system)

c is a pointer with 4 B size in 32 system and 8 B size in 64 system.

count isn’t a local variable but an input argument that should be reserved in registers.

Prog3:

a)

Prog3_32:
i:8.6 kB
ii: 437552 Bytes
iii: 432 Bytes

Prog3_64:
i: 10 kB
ii: 4413208 kB
iii: 520 kB

b)

valgrind results:

```
==4344== Argument 'size' of function malloc has a fishy (possibly
negative) value: -294967296
==4344==    at 0x402B5C8: malloc (vg_replace_malloc.c:309)
==4344==    by 0x804856C: allocate (prog3.c:13)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x8048619: main (prog3.c:31)
==4344==
==4344== Invalid write of size 1
==4344==    at 0x40324E0: memset (vg_replace_strmem.c:1253)
==4344==    by 0x80485A0: allocate (prog3.c:18)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x8048619: main (prog3.c:31)
==4344==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==4344==
==4344==
==4344== Process terminating with default action of signal 11
(SIGSEGV)
==4344==  Access not within mapped region at address 0x0
==4344==    at 0x40324E0: memset (vg_replace_strmem.c:1253)
==4344==    by 0x80485A0: allocate (prog3.c:18)
==4344==    by 0x80485DE: allocate (prog3.c:22)
```

```
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x80485DE: allocate (prog3.c:22)
==4344==    by 0x8048619: main (prog3.c:31)

==4344== HEAP SUMMARY:
==4344==     in use at exit: 444,444,440 bytes in 8 blocks
==4344==   total heap usage: 8 allocs, 0 frees, 444,444,440 bytes
allocated


Heap is fully allocated due to :
    b = malloc(pow(10,r)*sizeof(int));
at eighth term: arg of malloc will be 4*10^8



According to the printed results:
Stack Address = 0xffffcc44     Heap Address = 0x804b008

Stack Address = 0xffffca74     Heap Address = 0x804b038

Stack Address = 0xffffc8a4     Heap Address = 0x804b1d0

Stack Address = 0xffffc6d4     Heap Address = 0x804c178

Stack Address = 0xffffc504     Heap Address = 0xf7d36008

Stack Address = 0xffffc334     Heap Address = 0xf7965008

Stack Address = 0xffffc164     Heap Address = 0xf533f008

Stack Address = 0xffffbf94     Heap Address = 0xdd5c6008

/proc results:For limit of heap
0804b000-0806c000 rw-p 00000000 00:00 0
[heap]
The boundary of heap has already been busted at the 5th iteration

Then we know that heap(and whole mem) is fully allocated and even
extend to the address 0x0 at the eighth iteration.
```

c) Prog2 is caused by stack overflow, this is caused by heap overflow, but they are similar caused by mem explosion.

Prog4:

a):

```
==20307== HEAP SUMMARY:
==20307==     in use at exit: 1,843,002,000 bytes in 921,501 blocks
==20307==   total heap usage: 1,009,999 allocs, 88,498 frees,
2,039,998,000 bytes allocated
==20307==
==20307== LEAK SUMMARY:
==20307==    definitely lost: 1,842,952,000 bytes in 921,476 blocks
==20307==    indirectly lost: 0 bytes in 0 blocks
==20307==      possibly lost: 50,000 bytes in 25 blocks
==20307==    still reachable: 0 bytes in 0 blocks
==20307==         suppressed: 0 bytes in 0 blocks
```

Cause :Allocation of heap without freeing them in allocate()

only release mem by free() when func(i) = 1;

The func is a func to determine wether i is a prime number. If i is a prime, the func(i)will return 1 otherwise 0.

We know that 1~10000 aren't always primes, so there were many allocated spaced that were not freed.

Easy to fix it. Replace the flag of "if(func(i)) " by "if(true)". Freeing mem is base requirement

b)

i)User cpu time: 1.79168

ii) system cpu time 0.474334 s

user cpu time is the time that cpu is running user's program/app

system cpu time is the time that cpu is running on kernel tasks

iii)maximum resident set size: 1814736 Byte

signal received:0(since there is no net or socket connections)

iv)voluntary context switch: 3

v)involuntary context switch :10

Prog5:
fork:
exec:

wait:

kill:

exit: