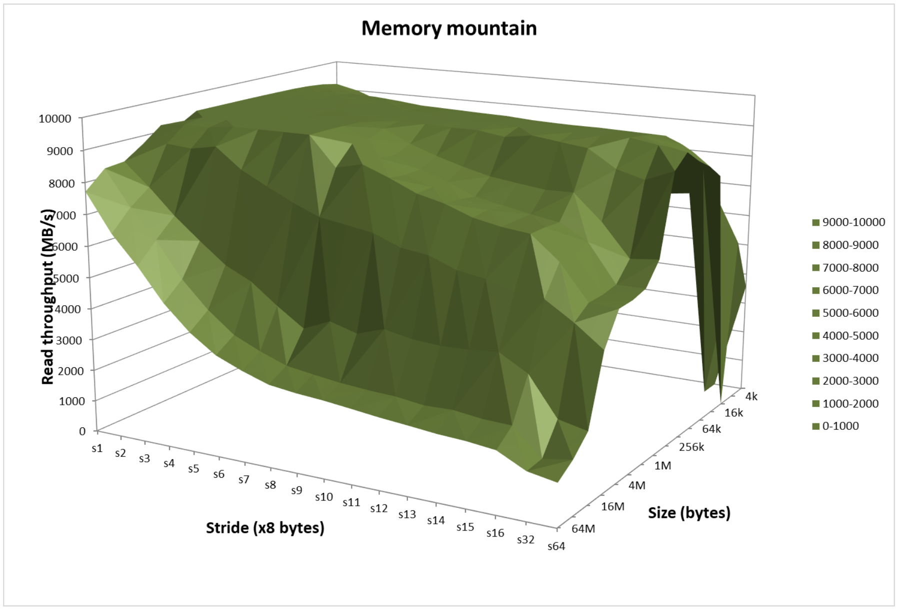
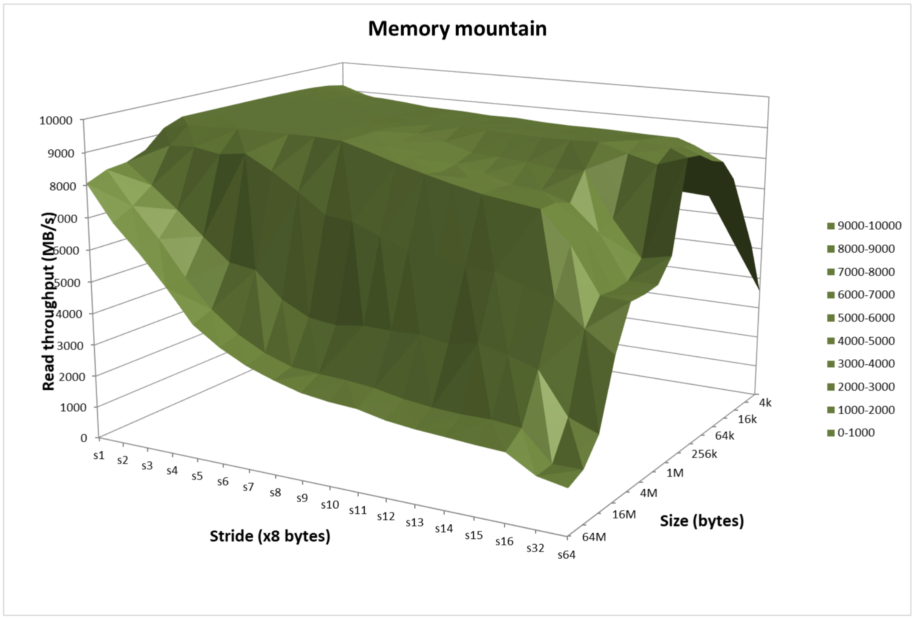
Noah Hanks

ECEN 424 HW 7

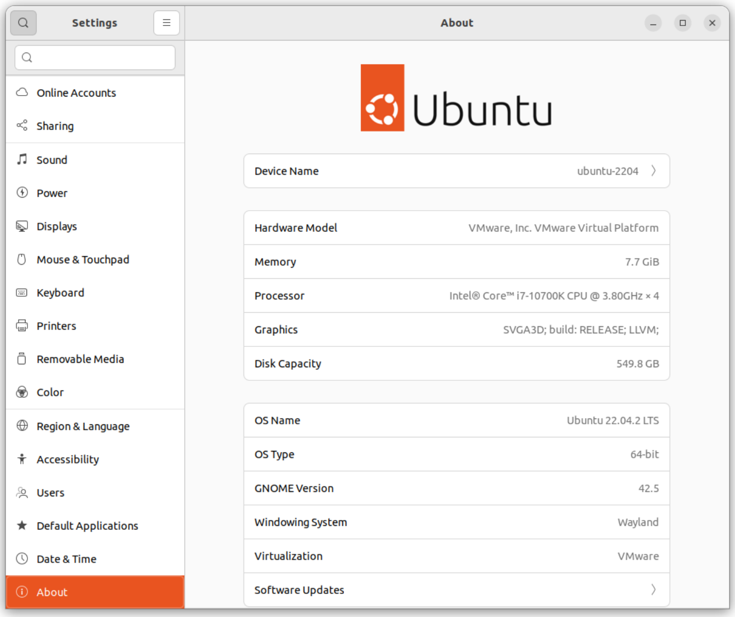
424-8)
Compiled using -O1



Compiled using -O3



Linux VM Specs



Device Name	ubuntu-2204
Hardware Model	VMware, Inc. VMware Virtual Platform
Memory	7.7 GiB
Processor	Intel® Core™ i7-10700K CPU @ 3.80GHz × 4
Graphics	SVGA3D; build: RELEASE; LLVM;
Disk Capacity	549.8 GB
OS Name	Ubuntu 22.04.2 LTS
OS Type	64-bit
GNOME Version	42.5
Windowing System	Wayland
Virtualization	VMware
Software Updates	

When using a higher optimization flag, the memory mountain had a higher read throughput value of a couple hundred. It seemed that with better optimizations, the overall performance was slightly higher. The graph of memory usage on my Linux VM is very different from the one presented in the textbook. It shows a steep spike at one point, rather than a gradual rise. This may be due to the limited cache allocation of the VM. Also, the overall rate of data access was considerably lower. By analyzing the overall height of the memory mountain, one can determine the amount of memory being consumed by the system. This could be used with other factors to determine what kind of system is being used.

7.8)

A.

a) main.1
b) main.2

A.

a) UNKNOWN
b) UNKNOWN

A.

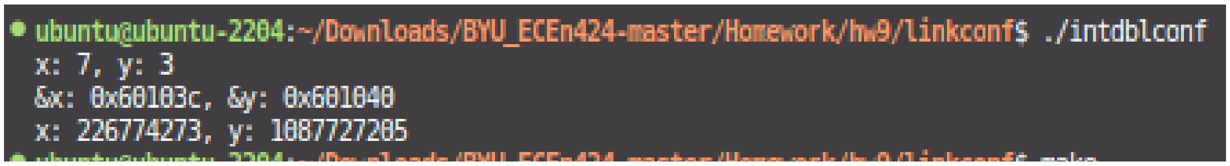
a) ERROR
b) ERROR

7.12)
A.
ADDR(s) = ADDR(.text) = 0x4004e0
ADDR(r.symbol) = ADDR(swap) = 0x4004f8
refaddr = ADDR(s) + r.offset = 0x4004ea
*refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr) = **0xa**

B.
ADDR(s) = ADDR(.text) = 0x4004d0
ADDR(r.symbol) = ADDR(swap) = 0x400500
refaddr = ADDR(s) + r.offset = 0x4004da
*refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr) = **0x22**

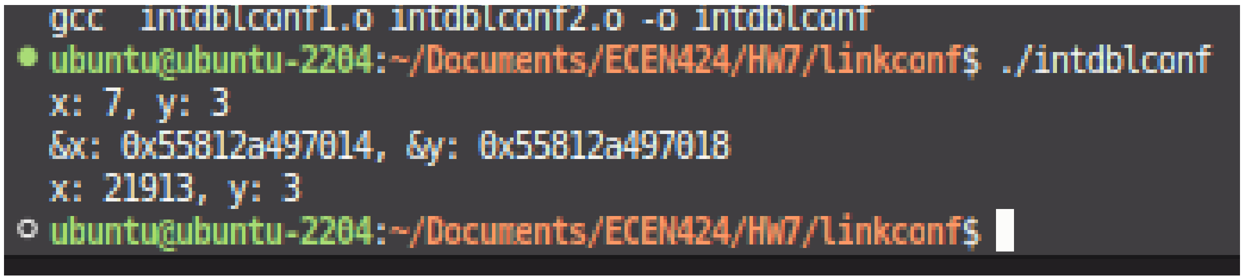
424-9)

Initial output:



```
● ubuntu@ubuntu-2204:~/Downloads/BYU_ECEn424-master/Homework/hw9/linkconf$ ./intdblconf
  x: 7, y: 3
  &x: 0x60103c, &y: 0x601040
  x: 226774273, y: 1087727205
```

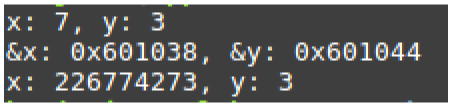
This bug is caused by the fact that p2() is called, it thinks that x is a double and assigns a double value to it which is too long and overflows into y.



```
  gcc  intdblconf1.o intdblconf2.o -o intdblconf
● ubuntu@ubuntu-2204:~/Documents/ECEN424/HW7/linkconf$ ./intdblconf
  x: 7, y: 3
  &x: 0x55812a497014, &y: 0x55812a497018
  x: 21913, y: 3
○ ubuntu@ubuntu-2204:~/Documents/ECEN424/HW7/linkconf$ 
```

I then changed intdblconf2.c to have the line “extern int x;” which caused the function p2() to instead truncate the double into an int.
Having link1 x as and int and link2 x as a double caused the linker to throw an error that said multiple definition of ‘x’. This remained whether I declared initial values for the variables or not.

Then I edited the line where x is declared in intdblconf1.c to int x;. When I did this there were the same warnings as the first time, but no error. This is the results:


```
x: 7, y: 3
&x: 0x601038, &y: 0x601044
x: 226774273, y: 3
```

Y is no longer being over written because the double declaration in intdblconf2.c is the strong symbol, so in main, x is a double. This way x is allocated large enough so that it can fit a double and y won’t get overwritten.