

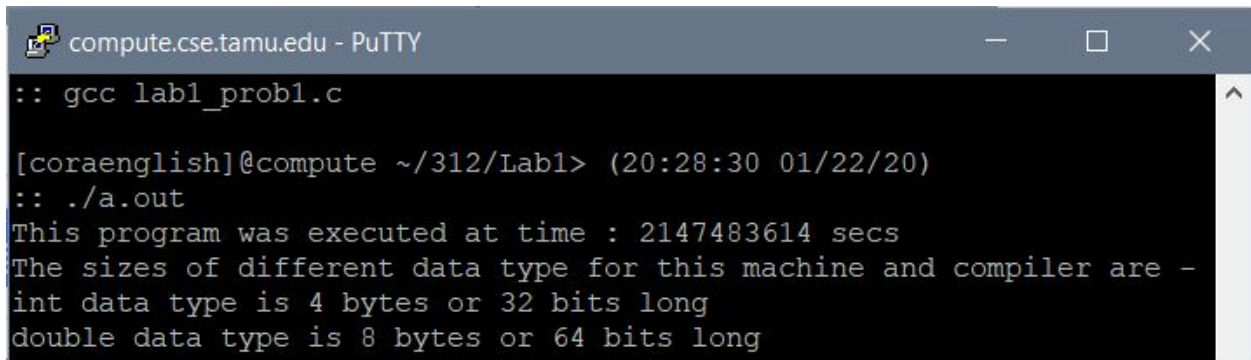
CSCE 312-201 LAB 1

Problem 1:

a) Using less than 4 sentences, explain what function/action, the two statements marked with “Tag 1” and “Tag 2” perform.

- Tag 1 creates a new integer variable called “int_var”. Tag 2 prints out the size of this integer variable using the `sizeof(variable)` function in bytes and then multiplies by 8 to convert to bits.

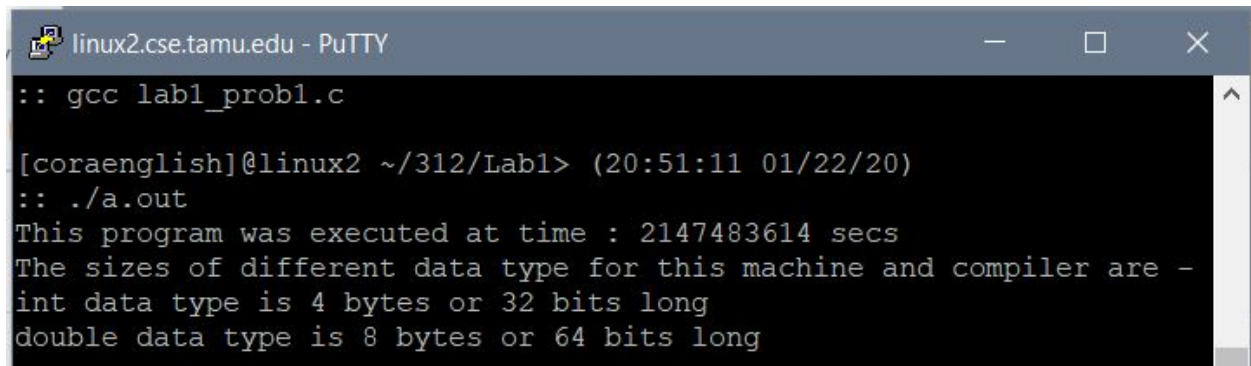
b) Compile and run this program on the CS CentOS 7 Linux machine (`linux.cse.tamu.edu`).



```
compute.cse.tamu.edu - PuTTY
:: gcc lab1_prob1.c

[coraenglish]@compute ~/312/Lab1> (20:28:30 01/22/20)
:: ./a.out
This program was executed at time : 2147483614 secs
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
```

c) Compile and run this program on the CS Red Hat Linux machine. (`compute.cse.tamu.edu`).



```
linux2.cse.tamu.edu - PuTTY
:: gcc lab1_prob1.c

[coraenglish]@linux2 ~/312/Lab1> (20:51:11 01/22/20)
:: ./a.out
This program was executed at time : 2147483614 secs
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
```

d) When you ran your code, did you get the time executed to be negative? If yes, why did that happen? (Since time cannot be negative). How could you fix this? Figure out how to fix it and do the necessary modifications.

- No, the answer was not negative. According to Pritam, on the older system, the answer would have been negative because of overflow. The system was updated so integers can now hold up to 4 bytes instead of the 2 bytes it could originally store so overflow is not an issue. A fix would have been changing it to a long instead of an int which is twice the size of int.

e) Change the data type of the variable `time_stamp` from double to long int. Re-run the program on both machines. Is there a change in the values reported? If so, which of the values is the correct value? Why is there a difference?

- No there was not a difference: 1579748389 secs when run with long int and 1579748402.00000 secs when run with double. Both values are correct because the `gettimeofday` function tells you how many seconds have elapsed since January 1, 1970 which is almost exactly 50 years ago. When you convert the seconds to calendar years, both results are a little over 50 years. Originally the double was not printing correctly but it was fixed.

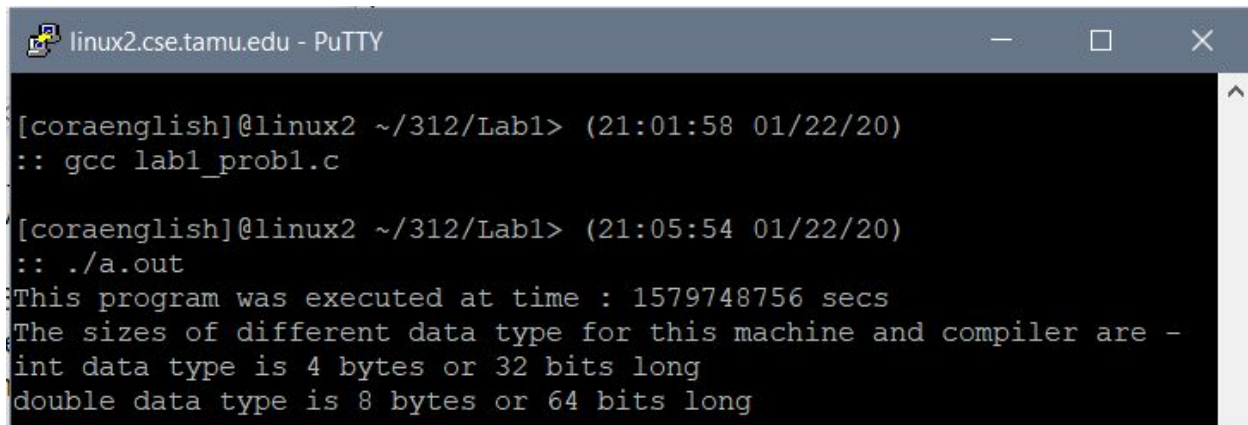
f) Find out the structure of type "timeval". Is it a standard C data type or is it platform specific?

- The `timeval` struct comes from the `sys/time.h` header which was pound included at the top of this program. It can be run on any platform as long as the header is included which makes it a standard C data type.

g) Submit your output files for all four runs.

For original versions see page 1. Fixed versions are below:

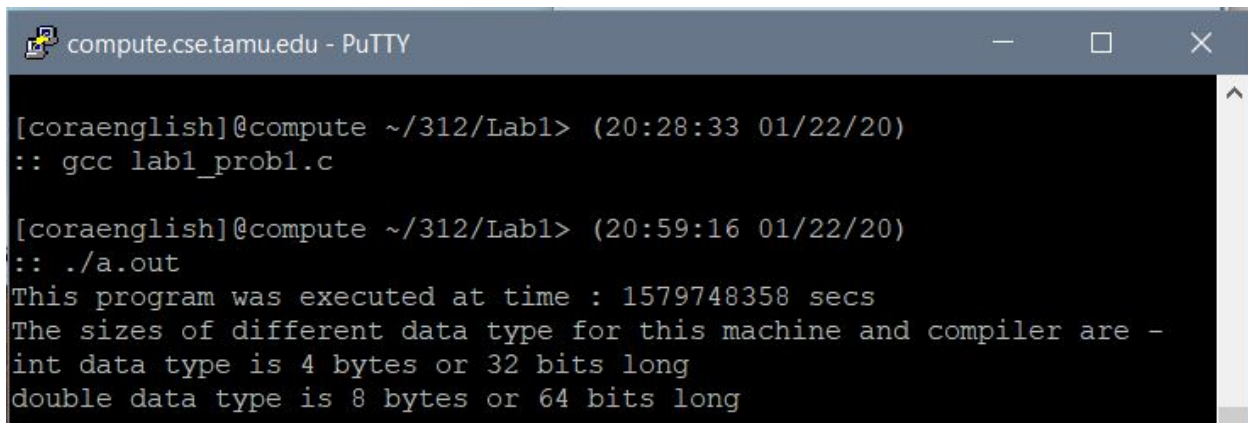
Linux with long int:



```
linux2.cse.tamu.edu - PuTTY
[coraenglish]@linux2 ~/312/Lab1> (21:01:58 01/22/20)
:: gcc lab1_prob1.c

[coraenglish]@linux2 ~/312/Lab1> (21:05:54 01/22/20)
:: ./a.out
This program was executed at time : 1579748756 secs
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
```

Compute with long int:

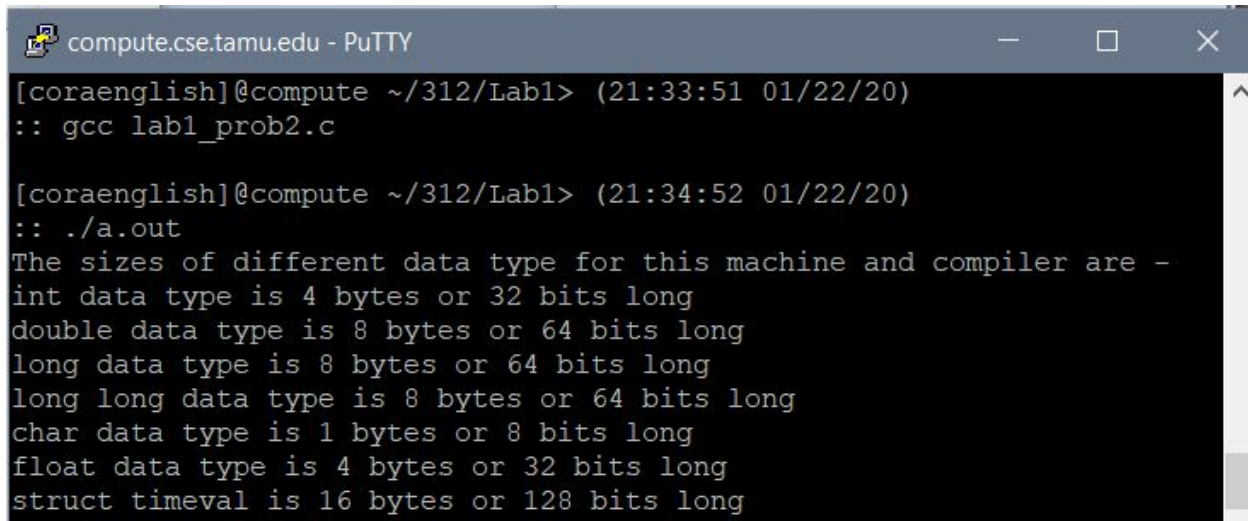


```
compute.cse.tamu.edu - PuTTY
[coraenglish]@compute ~/312/Lab1> (20:28:33 01/22/20)
:: gcc lab1_prob1.c

[coraenglish]@compute ~/312/Lab1> (20:59:16 01/22/20)
:: ./a.out
This program was executed at time : 1579748358 secs
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
```

Problem 2:

Output file can be found in zipped folder, snip of console is below:



```
compute.cse.tamu.edu - PuTTY
[coraenglish]@compute ~/312/Lab1> (21:33:51 01/22/20)
:: gcc lab1_prob2.c

[coraenglish]@compute ~/312/Lab1> (21:34:52 01/22/20)
:: ./a.out
The sizes of different data type for this machine and compiler are -
int data type is 4 bytes or 32 bits long
double data type is 8 bytes or 64 bits long
long data type is 8 bytes or 64 bits long
long long data type is 8 bytes or 64 bits long
char data type is 1 bytes or 8 bits long
float data type is 4 bytes or 32 bits long
struct timeval is 16 bytes or 128 bits long
```

Problem 3:

a) For each requirement, separately provide the Boolean expressions that you decided to use.

1. The BELL should chime/sound when the driver starts the engine without fastening his seatbelt.
 - $BELL = DOS \ \&\& \ !DSBF \ \&\& \ ER$
2. The BELL should sound when the driver starts the car without closing all the doors.
 - $BELL = DOS \ \&\& \ ER \ \&\& \ !DC$
3. The BELL should be off as soon as the conditions change to normal.
 - $BELL = DOS \ \&\& \ ER \ \&\& \ (!DSBF \ || \ !DC)$ (conditions for bell to be on, NOT this entire expression for bell off)
4. The doors should not lock when the driver has got out of the car but the keys are still inside the engine, even though the driver has closed the door lock lever. Note: If the driver is on the seat and requests the doors to be locked, the doors must lock.
 - $DLA = DLC \ \&\& \ (DOS \ || \ !KIC)$
5. The brake should be engaged when the driver presses the brake pedal. Brakes should disengage when the brake pedal is released. The brake should engage only when the car is moving, when the car is stationary the brake should not unnecessarily engage to reduce mechanical wear and tear of the brake's hydraulic system.
 - $BA = BP \ CM$

b) Create a single combined truth table with all the available sensor inputs and actuator outputs for all the five requirements together.

[illegible]

ALL OUTPUTS FOR 3 AND 4 CAN BE FOUND IN ZIPPED TXT FILES!

Problem 4:

a) Explain in a couple of sentences (with specific references to the code you wrote), where and how enum could help improve the readability and maintainability of your code.

- The enum given in the hint for problem 4 was extremely useful for testing the code before implementing the i/o system seen in the final product. Another place the enum class would be useful would be in the if statements. Instead of checking to see whether the bitmasked input was equal to its decimal counterpart, an enum could have been used. It would have worked very similarly to the hint given in the example code in the fact that the numbers needed (ex PATTERN_000000001 = 1) would be listed in the enum and attached to their decimal counterparts. So checking equivalence would be more visual and actually give you an idea of the inputs necessary for a certain output.

Problem 5:

a) Are you convinced that code developed for problem 4 is faster than the code developed for problem 3? Why or why not?

- No, I am not convinced that in this scenario the code developed for problem 4 is faster. This is because the measured time that was found using framework 5 varies from trial to trial, but this is in nanoseconds, not even in seconds. For some trials, problem 3 was faster, but in others problem 4 won. From the data I have after running many trials with the same inputs, I could not decide the clear winner. The computation complexity in each program is very low so the difference between them would not be big enough to make a difference. (Problem 3 vs. Problem 4)

```
compute.cse.tamu.edu - PuTTY
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4877 nanoseconds
The measured code took 0 seconds and 1850 nano seconds to run

[coraenglish]@compute ~/312/Lab1> (22:04:48 01/28/20)
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4493 nanoseconds
The measured code took 0 seconds and 917 nano seconds to run

[coraenglish]@compute ~/312/Lab1> (22:04:49 01/28/20)
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4932 nanoseconds
The measured code took 0 seconds and 894 nano seconds to run

compute.cse.tamu.edu - PuTTY
[coraenglish]@compute ~/312/Lab1> (22:06:42 01/28/20)
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4755 nanoseconds
The measured code took 0 seconds and 765 nano seconds to run

[coraenglish]@compute ~/312/Lab1> (22:06:43 01/28/20)
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4546 nanoseconds
The measured code took 0 seconds and 1103 nano seconds to run

[coraenglish]@compute ~/312/Lab1> (22:06:43 01/28/20)
:: ./a.out
Timer Resolution = 1 nanoseconds
Calibrartion time = 0 seconds and 4257 nanoseconds
The measured code took 0 seconds and 1553 nano seconds to run
```