Integers II
CSE 351 Autumn 2023

Instructor:
Justin Hsia

Teaching Assistants:
Afifah Kashif        Malak Zaki
Bhavik Soni          Naama Amiel
Cassandra Lam        Nayha Auradkar
Connie Chen          Nikolas McNamee
David Dai            Pedro Amarante
Dawit Hailu          Renee Ruan
Ellis Haker          Simran Bagaria
Eyoel Gebre          Will Robertson
Joshua Tan

http://xkcd.com/571/
Relevant Course Information

- hw4 due Monday, hw5 due Wednesday

- Lab 1a due Monday (10/9)
  - Use ptest and d1c.py to check your solution for correctness (on the CSE Linux environment)
  - Submit pointer.c and lab1Asynthesis.txt to Gradescope
    - Make sure you pass the File and Compilation Check – all the correct files were found and there were no compilation or runtime errors

- Lab 1b released today, due 10/16
  - Bit manipulation on a custom encoding scheme
  - Bonus slides at the end of today’s lecture have relevant examples
Runnable Code Snippets on Ed

- Ed allows you to embed runnable code snippets (e.g., readings, homework, discussion)
  - These are *editable* and *rerunnable*!
  - Hides compiler warnings, but will show compiler errors and runtime errors

- Suggested use
  - Good for experimental questions about basic behaviors in C
  - *NOT* entirely consistent with the CSE Linux environment, so should not be used for any lab-related work
Lesson Summary (1/2)

❖ Casting in C

▪ Data types determine size, interpretations, and operator behaviors
▪ Casting (implicit or explicit) can convert values between different data types
  • Be careful of the possible consequences of casting (truncation, zero/sign extension, change in interpreted value, change in operator behaviors like comparisons and shifting)

❖ We can only represent a limited range of numbers in $w$ bits

▪ When we exceed the limits, arithmetic overflow occurs following rules of modular arithmetic
  • Signed vs. unsigned overflow depends on interpretation of numbers

❖ Shifting is a useful bitwise operator

▪ Right shifting can be arithmetic (sign) or logical (0)
▪ Can be used in multiplication with constant or bit masking
Lesson Summary (2/2)

❖ Terminology:
  ▪ Modular arithmetic, arithmetic overflow (limits UMin, UMax, TMin, Tmax)
  ▪ Type casting: implicit vs. explicit, integer zero extension vs. sign extension
  ▪ Bit shifting: left shift, logical right shift, arithmetic right shift

❖ Learning Objectives:
  ▪ Identify when integer limitations are encountered (e.g., overflow).
  ▪ Identify the effect of C casts (both implicit and explicit) on stored values and the behavior of operations.

❖ What lingering questions do you have from the lesson?
Integers II – Context
Integer Representation Issues in Real Life

- **1985**: Therac-25 radiation therapy machine
  - Overdoses of radiation due to arithmetic overflow of incrementing a 1-byte safety flag variable

- **2000**: Y2K problem
  - Limited representation (two-digit decimal year)

- **2013**: Deep Impact spacecraft lost
  - Suspected integer overflow from storing time as tenth-seconds in unsigned int: 8/11/2013, 00:38:49.6

- **2038**: Unix epoch time rollover (seconds since 1/1/1970)
  - Signed 32-bit integer representation rolls over to Tmin in 2038
Discussion Question

❖ Discuss the following question(s) in groups of 3-4 students
   ▪ I will call on a few groups afterwards so please be prepared to share out
   ▪ Be respectful of others’ opinions and experiences

❖ Given that arithmetic overflow is a well-known property of integers in computing, what do you think are some of the causes and pressures that perpetuate these issues?
   ▪ Think broadly! Ideas could be technical, economic, societal, etc.
Integers II – Practice
Group Work Time

❖ During this time, you are encouraged to work on the following:
   1) If desired, continue your discussion
   2) Work on the lesson problems (solutions at the end of class)
   3) Work on the homework problems

❖ Resources:
   ▪ You can revisit the lesson material
   ▪ Work together in groups and help each other out
   ▪ Course staff will circle around to provide support
Practice Problems (1/2)

❖ What is the value (and encoding) of $T_{\text{Min}}$ for a fictional 6-bit wide integer data type?

❖ For unsigned char $uc = 0xA1$;, what are the produced data for the cast (unsigned short)$uc$?

❖ What is the result of the following expressions?
  ▪ (signed char)$uc >> 2$
  ▪ (unsigned char)$uc >> 3$
Practice Problems (2/2)

❖ Assuming 8-bit integers:
   - 0x27 = 39 (signed) = 39 (unsigned)
   - 0xD9 = -39 (signed) = 217 (unsigned)
   - 0x7F = 127 (signed) = 127 (unsigned)
   - 0x81 = -127 (signed) = 129 (unsigned)

❖ For the following additions, did signed and/or unsigned overflow occur?
   - 0x27 + 0x81
   - 0x7F + 0xD9

[TMin, Tmax] = [-128, 127]
[UMin, UMax] = [0, 255]