Khadichabonu Valieva

Dr. Sengupta

CSC 408

10 September 2024

Homework 2

**Problem 3**

* Short (usually 16 bits): **2^15 – 1** = 32767
  + Since one bit is reserved for the sign, a signed short can take value from -32768 to 32767
* Int (usually 32 bits): **2^31 – 1** = 2 147 483 647
  + Similarly, one bit is for the sign, so a signed int can take values from -2 147 483 648 to 2 147 483 647.
* Long (usually 32 or 64bits): (2^31 – 1) for 32 bits and **(2^63-1)** for 64 bits.
  + The size of long depends on what system is being used (32 or 64 bits)
* Long long (at least 64 bits): **2^63 – 1**
  + It’s guaranteed to be at least 64 bits on all systems allowing for very large integers.

**Problem 5**

char c = 'D'; // c holds ASCII value of D which is 68.

int x = c; // x is now 68

cout << x << endl; //prints value of x which is 68

int m = 8651266; // m holds that int value

short s = m; //short is typically 16 bits however 8651266 is a large int value. This causes overflow and truncation. The binary of 8651266 is 0100001000000001000000010 (24 bits), we take lower 16 bits and get 0000001000000010, which in decimal is **514**.

cout << s << endl; //prints 514

**Problem 6**

PART 1

short s1 = 705; //s1 holds 705

char c = static\_cast<char>(s1); //converts short value s1 to a char, 705 in binary 01011000001, however we take the last 8 bits which is **11000001** because of the char that holds values from 0 oto 255 (-128 to 127 for signed), and 705 exceeds the range. That 8-bits is **193** in decimal, so c is 193 now.

int i = c; //c is assigned to an int variable, which holds 193 now

cout << i << endl; // print i, which is 193.

PART 2

unsigned char c1 = s1; // 705 exceeds the range of unsigned char, so it takes the last significant 8 bits of c1=193 which is **11000001**

int i1 = c1; // i holds 193

cout << i1 << endl; //output 193