Lab Assignment #4

Purpose: understanding types and bitwise operations.
Collaboration: You have to work in a group of two.

Getting started

Change into your cse220 directory
Create a new directory called lab04
Change into the new directory
Implement the two programs below in your lab03 directory

Program 1

Write a program to extract the sign and the value of a negative integer:

Download lab4.c from the course website --> Lab Exercises. Copy it to your lab04 directory, compile and run. It will ask you for an input integer, and print out how that integer is stored as a binary representation in the computer. Run the program several times, try different values (e.g., 1, 2, 3, ...).
Run the program several times again, but now give it a negative integer as the input (e.g., -1, -2, -3, ...).
Pay attention to how these negative integers are represented.

Now let’s try to extract the sign (i.e., whether the integer is positive or negative, represented by the leftmost digit of its binary representation). Add the following three lines into the main function:

```c
unsigned short int mask = 0x8000;
printf("We define a mask as: ");
printBits(&mask, sizeof(mask));
```

Run the program again and see what the mask looks like (note that 0x8000 is a “hexadecimal” value, see the last problem of homework 3 for more information).

Further define a variable called sign:

```c
unsigned short int sign;
```

Now you need to use a bitwise operation to extract the sign of the input integer (which is stored in x), what bitwise operation should you use? Pick the correct bitwise operation to add the following statements into your code (replace which_bitwise_operation with the operation you picked):

```c
sign = x which_bitwise_operation mask;
```
if (sign > 0) {
    printf("%d is a negative integer.\n", x);
}

Run the program again and test it with some positive and negative values. If you’ve chosen the correct operation, it should be able to recognize the negative integers you input.

Now we want to get the value if the input is a negative integer, for example, if the input is -109, we want to output 109 as its (absolute) value. Declare one more variable:

    unsigned short int value;

Try the following code:

    value = ~x;
    printf("The complement of %d is: ", x);
    printBits(&value, sizeof(value));
    printf("The absolute value of %d is: %d\n", x, value);

Test the program with different negative values, what do you observe?

Wrap Up

You need to make one final change to the value variable so that we can get the correct result. Make that change and show your TA the result.

Program 2

Experiment with types and overflow.

Download lab4.2.c from the course website -> Lab Exercises. Copy it to your lab04 directory, compile and run. It defines two variables x and y. x is unsigned integer, so all the 16 bits is used to represent the value. y is signed integer, so the leftmost bit is used for representing the sign and the rest 15 bits for value.

How you can print out the maximum value that x can hold? (Hint: the maximum value of x is all its 16 bits are 1. Currently, the value of x is 0, so which bitwise operation you should use to turn all the bits into 1 ?)

Add these lines into your code (replace which_bitwise_operation with the operation in your mind):

    x = which_bitwise_operation x;
    printf("The value of x now is: %d\n", x);
Further add these lines:

```c
x += 1;
printf("The value of x now is: %d\n", x);
```

What happens here?
Show your TA the result and explain what you think happens here.

Next, how to get the maximum value of a signed integer variable (e.g., y)? (Hint: a signed integer variable holds the maximum value when its leftmost bit is 0 (i.e., positive) and all the remaining bits are 1.)

You can again use a mask:

```c
unsigned short int mask = 0x8000;
/* you need to make a change to the mask here */
printf("We define a mask as: ");
printBits(&mask, sizeof(mask));
```

and then do something with y and the mask, such like

```c
y = y & mask;
printf("The value of y now is: %d\n", y);
```

However, you need to make one more change to the mask before you can use `y = y & mask;`
How should you change your mask?

Make a change to the mask so that we can print the maximum value that y can hold.

Finally, add

```c
y += 1;
printf("The value of y now is: %d\n", y);
```

Show your TA the result and explain what happens again here.