CMPT 125: Week 1 Lab Work

Information Representation

1. Find the two's complement representations of the integers -125, 343, 128 and -128, 87, -87 as Bytes.

Solution

For -125,

- We first convert 125 to unsigned binary in Byte pattern to get **0111 1101**
- Then flip all the bits to get **1000 0010**
- Finally add 1 to the result to get **1000 0011. This is the required answer.**

For 343,

- We convert 343 to unsigned binary in Byte Pattern to get **1 0101 0111**. And that is our answer. **But wait a minute!** This has nine bits and the left most bit will be lost because of an overflow. Hence it will give rise to the binary **0101 0111** in a Byte pattern. This surely is not 343. In fact it is the decimal 87. Hence we conclude that we cannot store 343 in a Byte as two's complement; and if we attempt to do so what we get is 87.

For 128,

- We first convert 128 to unsigned binary to get **1000 0000**. And this is our answer. **But wait a minute!** This two's complement binary representation starts with a **1** and must be a representation of a negative decimal. In fact it is the representation of the decimal -128. Once again we conclude that we cannot store 128 in a Byte as two's complement; and if we attempt to do so what we get is -128.

For -128,

- We first convert 128 to unsigned binary in Byte pattern to get **1000 0000**
- Then flip all the bits to get **0111 1111**
- Finally add 1 to the result to get **1000 0000. This is the required answer.**

For 87,

- We convert 87 to unsigned binary in Byte pattern to get **0101 0111. This is the required answer.**

For -87,

- We first convert 87 to unsigned binary in Byte pattern to get **0101 0111**
- Then flip all the bits to get **1010 1000**
- Finally add 1 to the result to get **1010 1001. This is the required answer.**

2. Convert the following two's complement Byte patterns to decimal: **1000 0000, 1010 1001,** and **0101 0111**

Solution

For **1000 0000**,

- It starts with a 1 therefore it is negative decimal. Find its two's complement to get **1000 0000**
- Convert this Byte pattern to decimal to get 128. The required decimal is therefore -128.

For **1010 1001**,

- It starts with a 1 therefore it is negative decimal. Find its two's complement to get **0101 0111**
- Convert this Byte pattern to decimal to get 87. The required decimal is therefore -87.

For **0101 0111**,

- It starts with a **0**. Therefore it is positive decimal. Converting it to decimal by expanding it with powers of two gives 87. Hence we conclude that **0101 0111** in two's complement is 87 in decimal.

3. Perform the operation 125 – 87, -128 + 37 and -87 – 37 in two's complement using a Byte pattern and show that your answer is consistent with what we would expect if the arithmetic is performed in decimal.

Solution

For 125 – 87, we first write it as 125 + -87. Then we have

- 125 in two's complement in Byte pattern is **0111 1101**
- -87 in two's complement in Byte pattern is **1010 1001**
- Adding the binaries gives the result **1 0010 0110**
- The left most extra bit will be lost (overflow). Therefore the answer will be **0010 0110.** This binary corresponds to the decimal 38. In fact, performing the operation in decimals shows that 125 – 87 = 38. Hence we conclude that the arithmetic we performed using two's complement gives a correct answer.

For -128 + 37, we have

- -128 in two's complement in Byte pattern is **1000 0000**
- 37 in two's complement in Byte pattern is **0010 0101**
- Adding the binaries gives the result **1010 0101**
- Therefore the answer will be **1010 0101.** This binary corresponds to the decimal -91. In fact, performing the operation in decimals shows that -128 + 37 = -91. Hence we conclude that the arithmetic we performed using two's complement gives a correct answer.

For -87 – 37, we first write it as -87 + -37. Then we have

- -87 in two's complement in Byte pattern is **1010 1001**
- -37 in two's complement in Byte pattern is **1101 1011**
- Adding the binaries gives the result **1 1000 0100**
- The left most extra bit will be lost (overflow). Therefore the answer will be **1000 0100.** This binary corresponds to the decimal -124. To this end, performing the operation in decimals shows that -87 - 37 = -124. Hence we conclude that the arithmetic we performed using two's complement gives a correct answer.

4. Given the Byte binary 1101 0011. What value does it represent if it is
  a. Unsigned binary representation of an unsigned decimal number?
  b. Two's complement binary representation of a signed decimal number?
  c. Ascii code of a character?

5. **Challenge**

  a. Give the two's complement representation of -206 as a BYTE pattern.
  b. What signed decimal number does your BYTE in part (a) actually represent?
  c. Give the two's complement representation of 322 as a BYTE pattern?
  d. What signed decimal number does your BYTE in part (c) actually represent?
  e. Perform the binary addition of your answers for parts (a) and (c) and give your binary answer as a BYTE?
  f. Does the binary addition operation of part (e) give rise to an overflow?
  g. What signed decimal number does your BYTE in part (e) represent?
  h. Surprise surprise!!! Observe that your answer for part (g) is actually the correct sum of the signed decimal numbers -206 + 322. How did this happen? That is although your BYTE memories were not able to store the signed decimal numbers -206 and 322; the addition operation of the BYTEs however gave rise to correct answer. How?

Code Analysis Questions

1. What is the output of the following code snapshot assuming it is embedded inside a valid C++ program

```
float x = 3 + 5/7;
cout << x * 7 / 2 << endl;
```

2. What is the out of the following C++ code fragment assuming it is placed inside a valid C++ program.

```
int a = 5;
int b = 4;
int result = a / b;
cout <<  result << endl;
result = a * 1.0 / b;
cout <<  result << endl;
result = a + 1.0 / b;
cout <<  result << endl;
result = (a + 1.0) / b;
cout <<  result << endl;
result = a + (1.0 / b);
cout <<  result << endl;
result = a / 1.0 * b;
cout <<  result << endl;
result = a / 1.0 + b;
cout <<  result << endl;
result = a + 4 / b;
cout << result << endl;
result = a + b * (a – b)  / b % a;
cout << result << endl;
```

Programming Questions

3. Write a C++ program that reads in the height and base of a triangle and then prints the area of the triangle. You must decide what data types are appropriate for your variables.

4. Write a program that reads in the principal amount in a saving account, its interest rate, and the number of years since the account was opened. Your program then should calculate and print the total amount (i.e. the principal amount plus the interest) in the saving account after the time period specified. You must decide what data types are appropriate for your variables. Use simple interest.

5. Write a complete C++ program that reads two integer values d1 and d2 each of which is in the range [1, 30] representing calendar days of the same month from the user. Then your program must print the number of days from d1 to d2. For example if you enter 5 for d1 variable and 27 for d2 variable, then your program must print "There are 22 days from day 5 to day 27".

   Of course it is ok if your program prints the days as a negative number such as "There are -22 days from day 27 to day 5". The reason being the value of d1 might be larger than d2 and that will give a negative result. Assume a month has 30 days and that the input values for the days are valid; i.e. in the range [1, 30].

6. Write a complete C++ program that reads two integer values m1 and m2 each of which is in the range [1, 12] representing calendar months of the same year from the user. Then your program must print the

number of days from m1 to m2. For example if you enter 5 for m1 variable and 9 for m2 variable, then your program must print "There are 120 days from month 5 to month 9".

Of course it is ok if your program prints the days as a negative number such as "There are -120 days from month 9 to month 5". The reason being the value of m1 might be larger than m2 and that will give a negative result. Assume a month has 30 days and that the input values for the months are valid; i.e. in the range [1, 12].

7. Write a complete C++ program that reads two integer values y1 and y2 each of which is some positive number representing calendar years from the user. Then your program must print the number of days from y1 to y2. For example if you enter 1965 for y1 variable and 1974 for y2 variable, then your program must print "There are 3240 days from year 1965 to year 1974".

Of course it is ok if your program prints the days as a negative number such as "There are -3240 days from year 1974 to year 1965". The reason being the value of y1 might be larger than y2 and that will give a negative result. Assume a year has 360 days and that the input values for the years are valid; i.e. some positive numbers.

8. Write a C++ program that declares six variables named y1, m1, d1, y2, m2, and d2 all as integer data types. Now read the birth date of a child 1 in y1, m1 and d1 variables where y1, m1 and d1 represent the year, month and day of birth date of child 1. Then read the birth date of child 2 in y2, m2, and d2 variables. Finally print the number of days from the day child 1 is born until the day child 2 is born. It is ok if your program prints the number of days as negative or positive depending on the inputs (see questions 5, 6, and 7 above). Assume the input values for days are in the range [1, 30], the input values for months are in the range [1, 12], and the input values for the years are some positive numbers. This means a month has 30 days and a year has 12 months (= 360 days).

9. Write a program that asks the user to enter a non-negative integer in the range [0, 255] and prints the unsigned binary representation of the number in byte pattern. Assume the user will always enter a number in the range [0, 255]. Use modulo operator in order to make the computation easy.

Hint:- Declare eight **int** data type variables b1, b2, b3,...,b8 corresponding to the eight binary bits. You may assume b1 is the most significant bit and b8 is the least significant bit. Now compute the value of each of the bits and finally print the eight bit binary number. Because of the assumption made that b1 is the most significant bit and b8 is the least significant bit, this means the binary number must be printed in the order `cout << b1 << b2 << b3 << b4 << b5 << b6 << b7 << b8 << endl`.

For example, if the user input number is 153, then your program must print **10011001**

10. Answer Question number #9 above but this time without using modulo operator.

11. In Canadian currency, the available coin denominations are **toonie** (2 dollar coin), **loonie** (one dollar coins), **quarter** (25 cents), **dime** (10 cents), **nickel** (5 cents) and **penny** (1 cent). Write a C++ program that reads an amount of money from the user as a double data type (for example 17.69 to mean seventeen dollars and 59 cents) and prints the number of coins of each denomination such that the number of coins you need is the minimum among all possible combination of coins that give rise to the amount of money the user entered. For example an amount of money 17.69 must print 8 toonies, 1 loonie, 2 quarters, 1 dime, 1 nickel and 4 pennies.

12. Repeat Q8 above but this time your program must print the number of years, number of months and numbers of days between the birth dates of the two children. Assume a year has 12 months (= 360 days) and a month has 30 days. Your output must contain only positive numbers and possibly a zero OR only negative numbers and possibly a zero. NO MIX OF NEGATIVE AND POSITIVE OUTPUTS IS ALLOWED.

13. Write a C++ program that prints the lyrics of the **99 bottles of water on the wall** song onto the screen. The lyrics of the song is given below.

Ninety-nine bottles of beer on the wall,
Ninety-nine bottles of beer,
Take one down, pass it around,
Ninety-eight bottles of beer on the wall.

Ninety-eight bottles of beer on the wall,
Ninety-eight bottles of beer,
Take one down, pass it around,
Ninety-seven bottles of beer on the wall.

Ninety-seven bottles of beer on the wall,
Ninety-seven bottles of beer,
Take one down, pass it around,
Ninety-six bottles of beer on the wall.

⋮
⋮
⋮

One bottle of beer on the wall,
One bottle of beer,
Take one down, pass it around,
Zero bottles of beer on the wall.