Introduction to C
CS 2022 Spring 2011

Problem Set 3
Due Friday, 25 February 2011

---

1. **[Puzzle] Bitwise Operation [20 pts]**
   In this problem, you will implement the following functions according to their requirements. Each function has a list of legal operations, and a maximum number operations that you can use.

   - `int bit_and(int x, int y)`
     Returns `x & y`.
     Example: `bit_and(6, 5) = 4`.
     Max ops: 8
     Legal ops:

     `~ |`

   - `int bit_or(int x, int y)`
     Returns `x | y`.
     Example: `bit_or(6, 5) = 7`.
     Max ops: 8
     Legal ops:

     `~ &`

   - `int is_equal(int x, int y)`
     Returns 1 if `x == y` and 0 otherwise.
     Example: `is_equal(5, 5) = 1, is_equal(4, 5) = 0`
     Max ops: 5
     Legal ops:

     `! ~ & ^ | + << >>`

   - `int logical_right_shift(int x, int n)`
     Does a logical right shift of `x` to the right by `n`.
     Example: `logical_right_shift(0x87654321, 4) = 0x08765432`
     Max ops: 16
     Legal ops:

     `~ & ^ | + << >>`

   Download code template `bit_op.zip` from CMS [1]. The only file you need to edit is `bit_op.c`. I provide a `Makefile` for you to compile the code. There are also test cases in `bit_op.c`, you should not edit them. When you are done, compress the files again and submit the zip file `bit_op.zip` to CMS.

[1] `https://cms.csuglab.cornell.edu`

2. [**I/O**] **File I/O** [**40 pts**]
There is a very useful Unix tool called `cat`. It is used to concatenate files. If there are command-line arguments, they are interpreted as filenames, and processed in order. If there are no arguments, the standard input is processed. You need to print error message when files are not found. In this problem, you will implement this tool. Write your code in "`cat.c`" and submit it to CMS. If you compile your code to "`cat`", you will expect it has the same behavior as the Unix tool "`cat`".

3. [**Multithreading**] **Threads** [**40 pts**]
Recall that in assignment 1, we have evaluated the performance of a `C` program, which outputs any vector pair whose cosine similarity is greater than or equal to a user-defined threshold. In this problem, you will write a multi-threaded version. Download `calc.c` and `vectors.bin` from CMS first. `vectors.bin` is a **binary file** storing a 2 dimensional double array row by row. There are 10000 vectors in this file and each vector has 100 dimensions. Your program should read the data from `vectors.bin` first and create at least two threads to speed up the all pairs computation. You should also write a `Makefile` to manage your project. The program will output the number of vector pairs that pass the threshold and the running time directly. Finally, compress your `calc.c` and `Makefile` to `code.zip` and submit it to CMS.

4. [**Bonus**] **More on Bitwise Operations** [**20 pts**]
Implement the following functions.

   - `int least_bit_pos(int x)`
     Returns a mask that marks the position of the least significant 1 bit of `x`.
     Example: `least_bit_pos(96) = 0x20`
     Max ops: 30
     Legal ops:

     ```
     ! ~ & ^ | + << >>
     ```

   - `int sat_add(int x, int y)`
     Adds two values and if the result ($x + y$) has a positive overflow it returns the greatest possible positive value (instead of getting a negative result). If the result has a negative overflow, then it should return the least possible negative value.
     Example: `sat_add(0x40000000,0x40000000) = 0x7fffffff`
     `sat_add(0x80000000,0xffffffff) = 0x80000000`
     Max ops: 30
     Legal ops:

```
    ! ~ & ^ | + << >>
```

Download `bonus.zip` from CMS, in which `bonus.c` is the file you need to edit. There are test cases in the `main` function. You only need to implement the functions. Finally, submit your `bonus.zip` to CMS.