CS 100 Systems Programming
Homework 3

- Linux (30 pts): vi or emacs
  - (15 pts) Give the vi or emacs commands to do each of the following edits. State explicitly what editor you use. You are editing file which contains only one line:

    'Homer Simpson is intelligent.'
    - move to the beginning of the word 'intelligent' and change it to `an idiot`
    - mark where you are then insert the string '"IF" ' at the beginning of this line
    - move back to your mark then move to the end of the current word and append ' savant'
    - move back to the F in "IF" and change it to an f
    - insert a blank line before this line, write your file, and exit
  - (5 pts) Define an **insert macro** called \fo that inserts a C++ for loop with curly braces (indented as shown below) and leaves the user in insert mode just before the first semicolon (note the '>' is a greater-than sign indented by one tab):
    ```
    for ( ; ; )
    {
            >
    }
    ```
  - (5 pts) Define a **command macro** called !S that replaces all occurrences of <tab> in the file with four <spaces> (use <tab> and <space> to represent those characters).
  - (5 pts) Write a correct Makefile for building all your programs for this week. Do not use any default rules.
- C++ (30 pts): Write a String class which will be a wrapper class to the C style strings. The strings will be of varying lengths and must grow and shrink as necessary. Your String class must implement all the appropriate methods including constructors, assignment, equality operators, the index operator, reverse, indexOf (find), print, and read. I would prefer you not use any of the C *str* functions (e.g. strcmp, or strcpy), however you may write them yourself, as static methods, then use them. Here is String.h. Write the method definitions in String.cpp and your main in a file named testString.cpp.
  ```
  class String
   {
   public:
     /// Both constructors should construct
     /// this String from the parameter s
     String( const char * s = "");
     String( const String & s );
     String operator = ( const String & s );
  ```

```
    char & operator [] ( int index );
    int size();
    String reverse(); // does not modify this String
    int indexOf( char c );
    int indexOf( String pattern );
    bool operator == ( String s );
    bool operator != ( String s );
    bool operator > ( String s );
    bool operator < ( String s )
    bool operator <= ( String s );
    bool operator >= ( String s );
    /// concatenates this and s to return result
    String operator + ( String s );
    /// concatenates s onto end of this
    String operator += ( String s );
    void print( ostream & out );
    void read( istream & in );
    ~String();
  private:
    bool inBounds( int i )
    {
      return i >= 0 && i < len;
    }
    char * buf;
    int len; // the number of chars in buf
  };
  ostream & operator << ( ostream & out, String str );
  istream & operator >> ( istream & in, String & str );
```

  - Write a main function which tests each function defined in your class String.

- Systems Programming (40 pts): Write two C++ programs as follows
  a. (20 pts) Unix process creation: Write a program that uses fork() to create a total of 4 processes - each printing one letter of the alphabet from A to D 10,000 times. Have each process print it's process id (PID) when it is created. Also have each process flush the output after printing each character. Read about fork() in man or google. Be sure you do not leave any processes running. You can check this with the command, ps -aux. Note a parent process must wait() for each child after it has terminated, or that child will become a zombie.
  b. (20 pts) UNIX I/O Exercise: The purposes of this assignment are (1) to become familiar with UNIX I/O system calls and (2) to see the difference in performance between C++ library functions and system calls. The program you write should do the following.
     - Contain a main program that opens a file \`input\` for reading and a file \`output\` for writing. Make these files be command-line parameters to the program.

    - Implement a function that uses the in.get(char) and out.put(char) to copy the input file to the output file one character at a time.
    - Implement a function that uses the "UNIX" system calls read() and write() to copy the input file to the output file one character at a time.
    - Implement a function that uses the "UNIX" system calls read() and write() to copy the input file to the output file one buffer at a time (the buffer should be of size "BUFSIZ", which is declared in the <stdio.h> include file).
    - Once these functions are implement, test them out by having them perform their work N times each (where N is a command-line argument). Display the the amount of wallclock, user, and system time it takes for each of the four functions to execute.
    - Use a Timer class to measure the performance of each of the three approaches to copying a file. Use /usr/share/dict/linux.words for the input file to copy. I will provide a Timer class you can use.
- Your homework will be graded on *well.cs.ucr.edu*, so make sure it works on that machine.
- For *all* files use the following header:

```
/*
* Course: CS 100 Fall 2013
*
* First Name: <your first name>
* Last Name: <your last name>
* Username: <your username>
* email address: <your UCR e-mail address>
*
*
* Assignment: <assignment type and number, e.g. “Homework #2”>
*
* I hereby certify that the contents of this file represent
* my own original individual work. Nowhere herein is there
* code from any outside resources such as another individual,
* a website, or publishings unless specifically designated as
* permissible by the instructor or TA. */
```