# CMSC 100 – Processing Catchup and Makeup ExamFall 2013

The purpose of this assignment is to strengthen students’ Processing programming skills and to give them an opportunity to “earn back” some of the points that were lost on the midterm. The assignment has two phases: (1) problem solving practice and (2) a makeup exam.

**Problem solving practice:** You should spend time between now and the makeup exam exercising your “Processing muscles” by writing ten short programs, which are specified below. You may work individually, in groups, with TF/PM/instructor coaching – basically anything goes. If you’re stuck, you can feel free to email any of the course staff to ask for help (but please don’t post any complete or partial solutions on Piazza). However, you are doing yourself a disservice if you get so much help writing the programs that you don’t understand them and couldn’t write them on your own. You *do not* need to submit these programs – you don’t even have to do them if you don’t want to.

**Makeup exam:** On **Wednesday, November 6**, there will be a makeup exam session, starting at 5:30pm. (You ***must*** tell me whether you are planning to take the exam, no later than Friday, November 1. If you can’t arrive at 5:30pm, let me know at the time you tell me you’re planning to take the exam, and I will arrange to stay later. The exam will be in my lab, ITE 339, unless too many people sign up to take it, in which case I’ll arrange a room and let you know where it will be.) The makeup exam will be a 40-minute exam, worth 50 points. It will consist of 2 or 3 programming problems (identical or very similar to those in the practice problem list) and several multiple-choice questions about Processing. You will ***not*** be allowed to bring in a note sheet.

**Grade adjustments:** Your adjusted grade on the midterm will be:
 maximum (90, MidtermGrade + MakeupGrade – 10)
In other words, however many points you score above 10 on the makeup exam will be added to your midterm grade, *but* your final adjusted grade cannot be higher than a 90. For example, if you scored a 63 on the midterm and you score a 40 on the makeup exam, your adjusted midterm grade will be 90. If you scored a 52 on the midterm and you score a 35 on the makeup exam, your adjusted midterm grade will be 77. (If you scored 90 or higher on the midterm, there is no point in taking the makeup exam!)

**Practice problems (the first two should sound very familiar):**

1. Write a code segment, using a for or while loop, that calculates and prints the sum of the odd integers from 1 to 29, inclusive.
2. Write a function in Processing that takes two real-valued numbers (*height*, the height of a triangle, and *baseWidth*, the width of the triangle at its base), and returns the area of the triangle. (Hint: The area of a triangle with height *h* and width *b* is  .)
3. Write a function called *Change* that takes two parameters (*cost*, the amount of a transaction, and *cash*, the amount that a customer paid) and returns the amount of change that the customer should receive.
4. Write a code segment that calculates and prints the sum of all of the numbers between 0 and 1000 that are evenly divisible by 7.
5. Write a code segment that asks the user for a number (using the getIntInput function that we used in class – you can just use that as it was given to you, and you won’t have to know how to write it on any exam) and prints all of the multiples of that number from 1 through 10. For example, if the user enters 12, your program should print:
 12
 24
 ...
 108
 120
6. Write a function called *Factorial* that takes one parameter, an integer *n*, and returns *n!* (the product of 1 through *n*: *2!* is 2; *3!* is *6; 4!* is 24, etc.)
7. Write a function called *MuchLessThan* that takes two integer parameters, *x* and *y*, and returns *True* if *x* is at least 10 less than *y*. (For example, *MuchLessThan (12, 15)* would return *False*, and *MuchLessThan (1, 1000)* would return *True*.)
8. Write a code segment that computes and prints the total of an array of numbers. You can assume that the variable *length* has already been set, and the array *L* has already been initialized to contain *length* numbers. For example,
 int L = {1, 2, 3}; // initialize L to the list [1,2,3]
 int length = 3;
 << your code goes here >>
should print the total 6.
9. Write a code segment that computes the average of an array of numbers (where *L* and *length* have already been initialized as in the previous question).
10. Write a function called *CompoundInterest* that takes three arguments (*Principal* (an initial dollar amount), *Rate* (a percentage), and *Months* (the number of months)) and returns the final balance if the principal amount is compounded monthly at the rate *Rate*. This function should use a loop. (The way that compounded interest works is that at the end of each month, the new principal is equal to the original principal times one plus the interest rate. For example, *CompoundInterest (50, .05, 3)* should return 57.88, i.e., 50\*1.05\*1.05\*1.05.)