

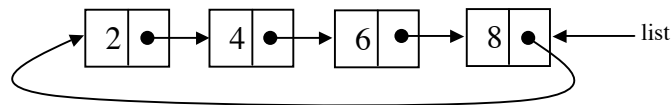
# Homework 1 - Due via Electronic Turnin Saturday, January 22 at 8:00 pm

59 points total

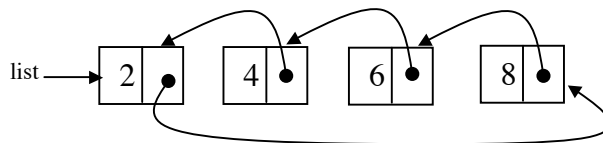
This homework will be turned in electronically. For the homeworks only you will need to turn in either a postscript file or a pdf (but this does not mean that you can't still use Word for creating a postscript file). It would be wonderful if you did not do the homework in Word but rather learned LaTeX or DIA, however, Word is just fine as well. Go to [http://www.cs.ucr.edu/cs14/cs14\\_04win/tutorials/postscript.html](http://www.cs.ucr.edu/cs14/cs14_04win/tutorials/postscript.html) for information on creating postscript and pdf files. DO NOT wait until the last minute to create your ps/pdf because late homeworks will not be accepted no matter what. Please make sure that you view/print your ps/pdf BEFORE you turn your homework in. Corrupted ps/pdf files WILL NOT be allowed to be resubmitted.

For 10% extra credit, form a study group of at least 3 people. Be sure to include the names of your partners and how long you worked on the homework together. Remember that you must work with at least 2 other people to form a group of three and you must work for at least 2 hours together (minimum of 1 hour sessions). See the main course web page for restrictions on homework collaboration.

- (8pts) In your textbook: page 197, exercise R-4.14. Make sure your algorithm is general. Do not assume a specific itemtype.
- (12 pts) Imagine a circular linked list of integers that are sorted into ascending order as show below:



The external pointer list points to the last node, which contains the largest integer. Write a function that revises the list so that its data are sorted into descending order as shown below:



Do not allocate or deallocate nodes.

- (12 pts) A polynomial can be represented as a linked list, where each node contains the coefficient and the exponent of the term. For example:



Write an algorithm to add two polynomials which may be of different lengths and can have any exponents (the two polynomials will not necessary only have the same exponents). You may assume that the exponents are ordered from highest (left/towards head) to lowest (right/towards tail). Assume there is a class called polynomial and three objects of that type already exist: P1 and P2 are the polynomials to add and P3 will be the sum of the two. State any other assumptions you make about the structure of the list (i.e. doubly linked, singly linked/tail pointer/sentinal/etc).

- (12 pts) Let A and B be two programs that perform the same task. Let  $t_A(n)$  and  $t_B(n)$ , respectively, denote their running times. For each of the following pairs, find the range of n values for which program A runs faster than program B. Specify the Big Oh notation for each function.

a)  $t_A(n) = 1000n$

$t_B(n) = 10n^2$

b)  $t_A(n) = 2n^2$

$t_B(n) = n^3$

c)  $t_A(n) = 2^n$

$t_B(n) = 100n$

d)  $t_A(n) = 1000n \log_2 n$

$t_B(n) = n^2$

5. (15 pts) Write a program that takes a single command line argument  $m$  (see the tutorials for information regarding command line arguments). The program should initialize a double variable  $x$  to 1, and then compute the following function  $2000^m$  times:  $x = x + (26.98 + 1)/26.98 - 1$ . Use the `pow` function in the math library to compute  $2000^m$ . (NOTE, you will need to include `cmath` and you will need to provide the command line option `-lm` to `g++` to link the math library) Run the program for  $m$  equal to 1 and 2 and measure the runtimes using the Unix “time” command (time executable name `arg1`). Run your program on a local PC in lab. Based on these runtimes, **estimate** the runtime for  $m = 3$  (**do not** run your program for  $m = 3$ ). This example demonstrates the tremendous differences between algorithms of complexity  $n$ ,  $n^2$ , and  $n^3$ . Paste a copy of your source code into your homework. Include your measured runtimes for  $m$  equal to 1 and 2 and your computation and final estimate of a runtime for  $m = 3$ .