ECE 1320 Optimization Methods
Winter 2003

Homework 1: Due in class Thursday January 16 2003

- This test contains 6 problems. They allow you to earn 100 points.
- Show your work, as partial credit can be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. **Be neat**.
- **No late submissions will be accepted.**
- Only homework returned in a 9in $\times$ 12in envelope will be accepted. (If you cannot find such envelope, ask the Instructor.) Please, write your name and the class name (ECE 1320) on the envelope (write clearly, please).

Write your name here: ______________________________

- **Problem # 1 [10 points]**. Write a (C++) function *piccolo* that given as input an array of integers numbers and the number n of the elements of the array (called the array *size*) returns the smallest among the array elements.

- **Problem # 2 [10 points]**. Write a (C++) function *oddSumma* that given as input an array of n integer numbers returns the sum of the "odd elements" of the array (i.e., the elements whose position is odd, such as the 1st element of the array, the 3rd, the 5th, and so on).

- **Problem # 3 [20 points]**. An array F of n numbers is called *fancy* if for each i, $1 \leq i \leq n$, the ith element equals the sum of all elements from the 0th to the $(i-1)$th (i.e., $F[i] = \sum_{j=0}^{i-1} F[j]$). Write a boolean (C++) function *isFancy* that given as input an array F of n integers determines if F is fancy or not.

- **Problem # 4 [20 points]**. A common problem for compilers and text editors is to determine if the parentheses in a strings are balanced and properly nested. For example, the string "((())())()" contains properly nested pairs of parentheses, while the string ")()(" does not. The string "())" does not contain properly matching parentheses. Write a C++ function *checkPar* that given as input a string s of parentheses returns *true* is s contains properly nested and balanced parentheses, *false* otherwise.

- **Problem # 5 [20 points]**. Write a C++ function *printSingles* that uses a one-dimensional array to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, print it out only if it is not a duplicate of a number already read.

- **Problem # 6 [20 points]**. (*The sieve of Eratosthenes*). A prime number is a number that is evenly divisible only by itself and 1. The sieve of Eratosthenes is a method of finding prime numbers. It operates as follows. a) Create an array of Booleans all initialized to *true.* Array elements with prime subscripts will remain *true.* All other array elements will be set to *false.* b) Starting with array subscript 2 (subscript 1 is not prime), every time an array element is found whose value is *true* iterate through the remainder of the array and set to *false* every element whose subscript is a multiple of the subscript for the element with value *true.* For instance, for array subscript 2, the array elements whose subscripts are multiple of 2 (i.e., 4, 6, 8, ...) will be set to *false*, and so on. When this process is complete, the array elements that contain *true* indicate that the subscript is a prime number. Write a C++ function *sieve* that given as input a number $n \geq 2$ prints out all prime numbers up to n.