CSci 2300 — Data Structures and Algorithms  
Week 1, Class 1 — January 9, 2001  
Introduction

Discussion of Syllabus

- Classes and homework (20%)
- Labs (10%): students must go to their scheduled labs
- Projects (25%) and late policy
- Exams (45%)
- Academic integrity
- Web site
- Schedule

We will start with a review of important concepts in C++.

Introductory Problem — Statistics of Primes

King Midas, a bad dude with lots of gold, doesn’t believe the number theorists. He doesn’t believe that there are approximately the same number of primes ending with 1, 3, 7 and 9. He wants empirical proof. He wants us to write a program that finds all primes up to size $n$ and count how many end with each digit. If we succeed, we will be rich and will have proven our worth to the king. The cost of failure is not so pleasant...

Algorithm — The Sieve of Eratosthenes

We begin our job by writing a function that takes an integer, $n$, and returns an array containing all of the primes up to and including $n$. This is based on an ancient algorithm developed by Eratosthenes, to be sketched in class.

Algorithm Implementation — Function `sieve_of_eratosthenes`

The implementation of this algorithm is appended to the end of the handout. We will go over the implementation and then use it as the backdrop for exploring a number of important concepts in C++. Most of this should be review. If any concept seems unfamiliar, review it on your own.
Dynamic Allocation of Arrays Using new and delete

- Pointer variables can point to individually allocated objects (memory locations) or to the first object in an array of allocated objects.
- Without initialization, the value of a pointer is undefined.
- Always draw pictures to visualize what’s happening with pointer variables.
- In the function, is_prime is defined as a pointer and initialized to point to a dynamically allocated array of bool variables:

  ```
  bool* is_prime = new bool[n+1];
  ```

- Arrays are deleted using lines of code such as

  ```
  delete [] is_prime;
  ```

  The square brackets indicate that an entire array is to be deleted.

- When we review linked lists we will see a context where individual objects are allocated and deleted instead of entire arrays.

Parameter Passing and References

- When a function needs to change a variable (object), the variable must be passed by reference. The variables prime_count and primes are reference parameters:

  ```
  void 
  sieve_of_eratosthenes( int n, 
  int& prime_count, 
  int*& primes)
  ```

- Remember: reference parameters “reference” the same memory locations as the variables passed to them in the function call.

- The syntax of passing a pointer by reference can seem confusing, so read it carefully. The int* indicates that a pointer is passed and the & indicates that it is passed by reference.

The Rest of the Code

We will quickly examine the rest of the code. Be sure to note:

- The test driver function test_sieve. The call to this function is “commented out” in the main program.

- The input of the integer from the command line.
• The `count_remainders` function. It counts the primes ending with 0, 1, 2, 3, etc. Note that `rcount` is treated as an array throughout, although it can also be thought of as a pointer.

• The deallocation of the `prime_numbers` array.

## Exercises and Reading

The code for `primes.cpp` is available through the course web page. You will may download it and play with it to solve the first three problems.

1. **(10 points)** Assuming the main program is changed to call `test_sieve`, what will be the output from function `test_sieve` if `prime_count` and `primes` are not reference parameters in function `sieve_of_eratosthenes`? Explain why this output occurs.

2. **(5 points)** The contents of the `rcount` array are changed in `count_remainders`. Why do we not have to make `rcount` a reference parameter?

3. **(5 points)** How much memory would be deleted in the function `sieve_of_eratosthenes` if we changed the line

   ```
   delete [] is_prime;
   ```

   to

   ```
   delete is_prime;
   ```

4. Read Sections 1.4 and 1.5 of the Weiss text.