```
int sum_array(const int a[], int n)
{
  int i, sum;

  sum = 0;
  for (i = 0; i < n; i++)
    sum += a[i];
  return sum;
}
```

7. Write the following function:

   ```
   bool search(const int a[], int n, int key);
   ```

   `a` is an array to be searched, `n` is the number of elements in the array, and `key` is the search key. `search` should return `true` if `key` matches some element of `a`, and `false` if it doesn't. Use pointer arithmetic—not subscripting—to visit array elements.

8. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variable `i` and all uses of the `[]` operator.) Make as few changes as possible.

   ```
   void store_zeros(int a[], int n)
   {
     int i;

     for (i = 0; i < n; i++)
       a[i] = 0;
   }
   ```

9. Write the following function:

   ```
   double inner_product(const double *a, const double *b,
                        int n);
   ```

   `a` and `b` both point to arrays of length `n`. The function should return `a[0] * b[0] + a[1] * b[1] + ... + a[n-1] * b[n-1]`. Use pointer arithmetic—not subscripting—to visit array elements.

10. Modify the `find_middle` function of Section 11.5 so that it uses pointer arithmetic to calculate the return value.

11. Modify the `find_largest` function so that it uses pointer arithmetic—not subscripting—to visit array elements.

12. Write the following function:

    ```
    void find_two_largest(const int *a, int n, int *largest,
                          int *second_largest);
    ```

    `a` points to an array of length `n`. The function searches the array for its largest and second-largest elements, storing them in the variables pointed to by `largest` and `second_largest`, respectively. Use pointer arithmetic—not subscripting—to visit array elements.

Section 12.4

W 13. Section 8.2 had a program fragment in which two nested `for` loops initialized the array `ident` for use as an identity matrix. Rewrite this code, using a single pointer to step through the array one element at a time. *Hint:* Since we won't be using `row` and `col` index variables, it won't be easy to tell where to store 1. Instead, we can use the fact that the first element of the array should be 1, the next `N` elements should be 0, the next element should