ES 112 Second Midterm Examination
Spring 2003-2004

1 Inverting a String In-Place

40 points

Write a C function that takes a string as an argument, and inverts its contents in-place. For instance, if the string “Hello, world!” is passed to the function, the function should modify it to become “!dlrow ,olleH”. The prototype of the function is:

```
void invert_string (char *s);
```

You are not allowed to assume a maximum size for the string, the function should work with strings of any size. Also, you are not allowed to allocate any memory using dynamic memory allocation.

2 Encrypting a File

30 points

One of the simplest (and easily crackable!) ways to encrypt a file is to take the file byte-by-byte, and XOR each byte in it with a single, fixed byte which we will call K (which we think of as the “key”). We will consider the bytes to be unsigned values in the range 0 to 255. Thus, obviously $0 \leq K \leq 255$. So, for encryption, we have:

$$F'(n) = F(n)\hat{}K$$

where $F'(n)$ is the n^{th} byte in the encrypted file, $F(n)$ is the n^{th} byte in the original file, and K is the encryption key.

Your task is to implement the following function:

```
void encrypt_file(const char *sourceFilename,
const char *destinationFilename, unsigned char key);
```

The parameter `sourceFilename` is the name of the source file, the parameter `destinationFilename` is the name of the destination filename, and the parameter `key` is the encryption key.

All possible errors should be detected and printed. An error should also be printed if the source and destination filenames are identical.

3 Random Number Generation

30 points

We wish to be able to pick a random point within the unit circle; i.e., a circle with radius one, and centered on the origin. Thus, we need to generate two random numbers x and y (which are the coordinates of the point. Thus, x and y must satisfy the relation:

$$x^2 + y^2 \leq 1$$

For this, we define

```
struct point {
    double x;
    double y;
};
```

and your task is to write the function

```
void generate_random_point(struct point *p);
```

which is supposed to fill in the elements of the parameter `p` with a pair of coordinates obeying the above conditions.

For this, you should use the `rand()` function. As a hint, the following will generate a random number between 0 and 1:

```
double x;
x = (double)rand()/RAND_MAX;
```

If you implement the function using any repetition structures, you will get 20 points. Correct implementation *without* using any loops will receive the full 30 points.