Name: _____   Precept (circle one): 1a   1b   2   3a   3b   4

COS 126 Fall 1996   **Oct. 10, 1996**
First Midterm Examination

Write your name and indicate your precept number on both pages of this exam. We'll separate the pages during grading, so your name must appear on every page. Also, please sign the pledge:

I pledge on my honor that I have not violated the honor code during this examination.

_____

1. (20 pts) Indicate the type and value of the expressions listed below, as illustrated by the first expression. Assuming the expressions are executed in the order shown and assuming the following declarations.

   ```
   unsigned u = 101; int i = -10; float f = 4.5; double d = 35.0;
   ```

type	*expression*	*value*
<u>int</u>	`i`	<u>-10</u>
_____	`-i+u`	_____
_____	`u%9`	_____
_____	`f+i`	_____
_____	`f*2.0+i`	_____
_____	`i++/3`	_____
_____	`-d+u*2`	_____
_____	`abs(-f-0.5)`	_____
_____	`i=d`	_____
_____	`(++i>0)+f`	_____
_____	`f+(u==i)*d`	_____

2. (10 pts) For each C integer constant listed below, give the equivalent value in the base indicated, as illustrated by the first line.

<u>12_8</u>	10 in octal
_____	1010 in hexadecimal
_____	043 in binary
_____	0256 in decimal
_____	0xF02C in octal
_____	43 in base 6

3. (5 pts) Give a one-sentence description of what the following code does to the array `x`.

   ```
   int i, x[20] = { 10, 11 };
   for (i = 2; i < 20; i++) {
           int k = rand()%i;
           x[i] = x[k];
           x[k] = i + 10;
   }
   ```

4. (20 pts) In the program below, identify the *scope* of each identifier by filling the blank to the right of its declaration with the lines on which it visible, as illustrated for f.

```
1     int x;
2     void f(int a, int b[]) {
3             x = a + b[0];
4             if (a < 0) {
5                     int a = x;
6                     int x = b[-a];
7                     while (x < 10)
8                             a += b[x++]; }
9             else if (a > 0) {
10                    int a = x + 1;
11                    x *= b[a] + a; }
12    }
```

x: ————————————————————————

f: <u>2–12</u> a: —————————— b: ——————————

a: ————————————————————————

x: ————————————————————————

a: ————————————————————————

5. (15 pts) `reverse(x, y, len)` copies `len` elements from `y` into `x` in reverse order. Here's an implementation:

```
void reverse(int x[], int y[], int len) {
        int i;

        for (i = 0; i < len; i++)
                x[i] = y[len-i-1];
}
```

This implementation has a serious bug when `reverse` is called with certain combinations of arguments. Given a one-sentence description of the bug, and rewrite the body of `reverse` so that it's correct.

6. (15 pts) Here's a fragment from `wf.c`, a program that prints the number of times its argument words appear in the input.

```
int main(int argc, char *argv[]) {
        char word[100];
        int i, counts[100] = { 0 };

        while (scanf("%s", word) != EOF) {
                ...
                if (i > 0)
                        counts[i]++;
        }
```

The code indicated by "`...`" searches for `word` in `argv` and sets `i` to the index in `argv` at which `word` occurs; otherwise, it sets `i` to 0. Write this missing code *without calling any functions, like* `strcmp`. Hint: an array of strings is like a 2-dimensional array.

7. (15 pts) `strcat(dst, src)` *appends* the string in `src` to the string in `dst`. Fill in the body of `strcat` below.

```
void strcat(char dst[], char src[]) {




}
```