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;
```

<table>
<thead>
<tr>
<th>type</th>
<th>expression</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>i</td>
<td>-10</td>
</tr>
<tr>
<td></td>
<td>-i+u</td>
<td></td>
</tr>
<tr>
<td></td>
<td>u%9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>f+i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>f*2.0+i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i++/3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-d+u*2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>abs(-f-0.5)</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>i=d</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(++i&gt;0)+f</td>
<td></td>
</tr>
<tr>
<td></td>
<td>f+(u=i)*d</td>
<td></td>
</tr>
</tbody>
</table>

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

```
128       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.

```c
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 }
```

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

```c
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.

```c
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.

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

}