Solutions to 211 Midterm 1998

1.

```
long Fib(short n)
{
     short i, j, k;
if(n==0 || n==1) return 1;
i= j=1; k=−1; /* k=−1 for safety : returns -1 if n≤ −1*/
while(n >1) {
     k=i+j;
     j=i;i=k; n--;
     }
return k;
}
```

2.

The issue here is precision: floats are fixed precision decimals. In binary (as in decimal) 1/3 is an infinite decimal, so truncating it produces errors. Let's do the example in decimal: 1.0/3.0 is a fixed length decimal, say .333 to have an explicit example. The second time through the loop index=.666, the third time index=.999, the fourth time index=1.332 and so on. Hence index is never 1.0 precisely and the loop never terminates.

3.

(A) and (B) compile and (C) does not. There is not much to say about why (A) and (B) compile, just that they are both legal C code. (Indeed they are functionally the same.) Program (C) fails to compile because you can not do a declaration that asks for an amount of space that the compiler can not determine. If for some reason you want to do the sort of thing that (C) does you need malloc!

4.

After line 1;
i=1; j=3; k=2;
after the statement i++; i=2; j=3; k=2;
after the statement j+=k; i=2; j=5; k=2;
after the statement k=i*j; i=2 ; j=5; k=10;
after the statement k11; i=2; j=5; k=9;
so the value of k after the indicated statements is 9.

5.

The function returns the short n so let's follow the value of n as we do the function. At the start, x=1.3; y=2.4; and n=6;

after the first line of code, tmp has some decimal value which we do not compute (at least yet). The next line takes n and adds n to it, so the new value of n is 12. Then we return 12 without ever using tmp. Hence at the end x has the value 12.0 (a float with value 12).

6.

```
i=0; y=1.2;
while( i < 33 ) {

        Some Code

    i+=2; x=x*x+1;
    }
```