Math 211 Midterm
March 27, 2002

Professor L. Taylor

Name: ____

1. The statements below represent part of a long program. The `while` loop was intended to execute some bit of code three times (with `index` being 0, $\frac{1}{3}$ and, $\frac{2}{3}$) and then move on. Assume that the code in **Some code** uses the value in `index` but does not change it.

Then in fact, the `while` loop repeats many more than three times and may very well be an infinite loop. Why?

```
    ⋮
float index;
    ⋮
index=0.0;
while(index!=1.0) {
        Some Code
        index+=1.0/3.0;
        }
```

. . . .

The issue here is one of precision. `index` is a floating point number so the representation of `1.0/3.0` is not exact: think finite decimal .3333 say. When we increment `index` 3 times it will be very close to 1.0 but not precisely equal so the test will remain true and the loop will continue.

It is possible that the loop will terminate eventually. Since there are only finitely many possible real numbers which can be represented in our machine, we might eventually luck into 1.0 and the loop would quit, but certainly not when we expect it.

2. The following `for` loop prints a sequence of integers on a line. Write those integers on the line below in the order in which they are printed. Assume `ii` has been declared already.

```
for( ii=1; ii<=30; ii*=ii,ii++) {
       printf("%d ",ii);
       }
```

Integers = ____

. . . .

First `ii` is set to 1 and the test condition is checked: since $1 \leq 30$ the test succeeds and we print a 1. Then we do the increment condition which is done in two steps: first we do `ii*=ii` which has the affect of replacing the value in `ii` by the square of that value. Next we increment the value in `ii` by 1. In our initial case `ii=1` so now it is $1^2 + 1 = 2$. Then we check the test condition: since $2 \leq 30$ the test succeeds and we print a 2. The increment conditions replace the value in by $2^2+1 = 5$. Again we check the test condition: since $5 \leq 30$ the test succeeds and we print a 5. The increment conditions replace the value in by $5^2 + 1 = 26$. Again we check the test condition: since $26 \leq 30$ the test succeeds and we print a 26. The increment conditions replace the value in by $(26)^2 + 1$ which is bigger than 30 so the for-loop quits.

3. Consider the following code for a function:

```
short strange(char cc) {
char xx=4;
switch(cc){
   case 'a': xx+=3;
   case 'b': xx%=3; break;
   case 'c': xx-=123;
   default: xx=0;
   }
return(xx);
}
```

What value is returned by `strange('a')`? _______

What value is returned by `strange('c')`? _______

Be sure to read the code carefully.

. . . .

This is an exercise in being careful. When evaluating `starnge('a')` we begin the `switch`-loop with `case 'a':`, which adds 3 to the value in `xx`, xo `xx` is now 7. Since there is no `break;` we drop through and do the next line, which replaces 7 by its remainder when divided by 3, so `xx` is now 1 and this is the value returned.

When evaluating `starnge('c')` we begin the `switch`-loop with `case 'c':` which subtracts 123 from the present value in `xx`. Again we drop through so this value is replaced by 0 and that is the value returned.

4. You have just solved a linear second order differential equation using the power series method. You have written your solution as $\sum_{n=0}^{\infty} a_n\, x^n$ and determined that the a's are related by the recursive relation $a_n = 2 * a_{n-1} - 3 * a_{n-2}$. You have also determined that $a_0 = 1$ and $a_1 = -3$. You want to examine the other coefficients so you write a program to print out the first few values. I have written such a program for you *except* you need to add some code to fill in the values in the `a`-array. You are to add your code in the blank space between the line initializing `aa[0]` and `a[1]` and the `for` loop doing the printing. A second copy of my code is provided *below* the horizontal line for your scratch work so your final answer should be legible.

```
#include <stdio.h>
#define   MAX   30

main(){
short ii;
short a[MAX];

a[0]=1; a[1]=-3;




for(ii=0;ii<MAX;ii++) {
   printf("a_%d = %d\n",ii,a[ii]);
   }
}
```

```
#include <stdio.h>
#define   MAX   30

main(){
short ii;
short a[MAX];

a[0]=1; a[1]=-3;




for(ii=0;ii<MAX;ii++) {
   printf("a_%d = %d\n",ii,a[ii]);
   }
}
```

. . . .

Add the following code after the initializations `a[0]=1; a[1]=-3;` and before the `for`-loop doing the printing:

```
for(ii=2;ii<MAX;ii++) {
   a[ii]=2*a[ii-1]-3*aa[ii-2];
   }
```