**Math 211 Midterm**
**March 23, 2001**
Professor L. Taylor

**Name:** _____

**1.** Suppose we have some function whose code begins with the following declarations and initializations.

```
int ix=1, iy=2;
int *ip=&ix;
```

1. If the next line reads
   `*ip=iy;`
   what values do both `ix` and `iy` have?

   ```
   ix = 2
   iy = 2
   ```

2. If *instead* of 1. the next line reads
   `iy=*ip;`
   what values do both `ix` and `iy` have?

   ```
   ix = 1
   iy = 1
   ```

3. If *instead* of 1. or 2. the next line reads
   `iy=(*ip)++;`
   what values do both `ix` and `iy` have?

   ```
   ix = 2
   iy = 1
   ```

4. If *instead* of 1., 2. or 3. the next line reads
   `iy=++(*ip);`
   what values do both `ix` and `iy` have?

   ```
   ix = 2
   iy = 2
   ```

**2.** A standard C idiom for reading standard input is

```
while( (c=getchar())!=EOF) {
    Some Code
    }
```

Rewrite this as a `for` loop.

```
for( c=getchar(); c!=EOF; c=getchar()){
    Some Code
    }
```

**3.** After the following declarations and code, what are the values of each of the variables? Be careful!

```
float a, b;
short i, j;
i=2; j=3;j+=i;
a*=j;
b=i; b-=j;
b+=0.03;
```

At the end of the first line after the declarations, `i=2` and `j=5;`
The second line after the declarations results in an undefined value being in `a`. It is `5` times the value that had been in `a`, but since that was undefined, so is the new value.
The second third after the declarations puts a `2.0` in `b` and then subtracts `5` from it, so the value then is `b=-3.0;`.

The last line adds `0.03` to `b` and then buts the answer back in `b` so `b=-2.97;`.

**4.** Write a function `Fact` which returns a long integer; takes a long integer as a variable and *recursively* computes the factorial function. You may assume that the function has been *declared* elsewhere. So `Fact(4)` should return the number `24` which is 4! and `Fact(5)` should return the number `120`, etc. Remember that 0!=1.

The factorial of a negative number is undefined but we must return something even when we have a negative number input, so let your function return the negative number that was entered and print NO error message of any kind.

```
long Fact(long LL) {
if(LL<0) {return(LL); }
else if(LL==0) {return(1); }
else {return(LL*Fact(LL-1)); }
}
```