**Practice with `#define`, `const`, `typedef`**

```c
#define PI 3.141592654   // usually use caps for #define name
.

float radius, area;

printf("Enter the radius for the circle: \n");
scanf("%f", &radius);

area = PI * radius * radius;
printf("The area of the circle is: %f \n", area);
```

Or, using a `const` is just as good. A `const` variable means the value will remain what it is initialized with (will remain constant), and cannot be changed.

```c
const double pi = 3.14159264;
float radius, area;

printf("Enter the radius for the circle: \n");
scanf("%f", &radius);

area = pi * radius * radius;
printf("The area of the circle is: %f \n", area);
```

**Difference between `#define` and `const`???**

```c
#define  NUMBER 5+2
int x = 3 * NUMBER;
printf("x is:  %i \n", x);    // what will be printed here???

const int number = 5+2;
int x = 3 * number;
printf("x is:  %i \n", x);    // what will be printed here???
```
#define is handled by the preprocessor; it is just simply a text replacement, a symbolic name for a constant. Before the actual compilation takes place, the preprocessor replaces all occurrences of the #define with whatever follows the #define name. Defines are often put at the top of the program, above any functions, or they can be placed in the functions themselves, as long as they appear before the #define name occurs. No matter where they are place, they can be used by any functions in the file (sort of like a global, except that a #define is not a variable).

**const** is a keyword that can be used with any variable declaration to force that variable's value to remain constant and not able to be changed. It is handled by the compiler, not the preprocessor. It appears before the type in the declaration:

```c
const int x = 4;
const char ch = 'a';
```

**typedef** allows you to create a synonym for a type. It doesn't create a new type, but a new name for an existing type. It is handled by the compiler, not the preprocessor. This is especially useful for things like structures and pointers, but since we haven't gotten to structures or pointers yet, here is an example with an int:

```c
typedef int Length;
```

This then allows you to declare things to be of type **Length** (which is really of type int):

```c
Length theLength, maximumLength;
```

The above example could be achieved using a #define instead:

```c
#define Length int
```

Another example of using typedef:

```c
typedef char LineBuffer[81]; // defines a type called LineBuffer
LineBuffer test, inputLine; // declares two char arrays, each of length 81
```

With this example, LineBuffer would not have been able to be defined with a #define.