Enumeration Types

Typically, enumerated types are used to enhance a program's readability. Consider the following declaration example:

```
typedef enum { North, East, South, West } directionT;
```

Once you do this, you can declare a variable of the type directionT:

```
directionT myHousedir = East;
direction yourHouseDir = West;

if (yourHouseDir == West) printf("Let's watch the sunset!");
```

Internally, any enumerated type declared this way has corresponding integer values stored, starting at 0. (Thus in our example, North is 0, East is 1, South is 2, and West is 3.) However, we can specify the internal integer values of enumerated types as follows:

```
typedef enum {
  penny = 1,
  nickel = 5,
  dime = 10,
  quarter = 25,
  halfdollar = 50
} coinT;
```

Why use enumerated types?
1) Each code/value does not need to be specified explicitly.
2) Code readability, which is also useful in debugging.

For our assignment we have:

```
typedef enum
{
      NONE = 0,
      GAS = 1,
      ICE = 2,
      STORM = 3,
      BARREN = 4,
      TEMPERATE = 5,
      LAVA = 6,
      OCEANIC = 7,
      PLASMA = 8,
      UNKNOWN = 9
} Planets;
```

And we can use this to increase readability in our assignment. We can just treat each member of the enum as a constant as in the following example:

```
switch (planet->type)
{
      case STORM:
            printf("STORM");
      default:
            printf("NONE");
}
```