Mark Remec-Pavlin <mremecpa@utk.edu>: undergraduate research.
Meeting on establishing an undergraduate research organization, next Tues, Oct. 26, 6:30, Haslam Business 203.

Exam I: class average was 58.
There were four exams in the 90s. Congratulations!
Go through your exam and make sure you understand what was wrong with your incorrect answers.
If it looks like we missed grading your correct answer correctly, please contact your TA or me.

Objects

In OOP (object oriented programming) we define classes of computer objects that often represent real or imaginary physical objects. In this case a playing card and later a deck of cards. We have to decide:

(1) What are the properties of the objects that we want to model in our program. For example, if we doing a computer card game, we need to know the suit and rank of the card, but we probably don't need to know its physical dimensions, back design, etc.

(2) We need to know any "behaviors" that the objects have (i.e., things that they can do) that are relevant to the program. So for a deck of cards, we might want to be able to print it, shuffle it, deal it, sort it, etc.

(3) We need to represent the objects in terms of data types that we already have. We have to encode the properties and behaviors in terms of things we already have: either things that are built into C++ (ints, chars, strings, doubles; +, -, <<,

==, !=, …), or things that we or somebody else has already defined.

Consider the Card data structure.
For simplicity assume its properties are suit and rank, and we can print them (and do several other things we'll add). An example declaration of Card:

```
struct Card {
  int suit;
  int rank;
  void print () const;
}
```

Encoding: We can represent the suit by an integer, for example, CLUBS = 0, DIAMONDS = 1, HEARTS = 2, SPADES = 3. This encodes the possible suits as integers.
I assigned the codes in alphabetical order, so that I could remember them better. Same thing with the rank, but what code is QUEEN? It's hard to remember.

Labeling Principle: Avoid requiring a user (or programmer) to remember a list of things more than few long, or to remember the position of something in a list. Instead, associate a meaningful label (name) with each possibility.

One way to do in C++ is to define constants:

```
const int CLUBS = 0;
const int DIAMONDS = 1;
const int HEARTS = 2;
const int SPADES = 3;
```

Then I can use CLUBS, etc. instead of the numerical code.

There is an even more convenient way to do this in C++, the enumeration type. It looks like this:

enum Suit { CLUBS, DIAMONDS, HEARTS, SPADES };

This assigns the code 0 to CLUBS, 1 to DIAMONDS, etc. For the ranks:

enum Rank { ACE=1, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING };

This assigns the codes in order, beginning with 1.

This is a way for declaring a type where you have small to moderate number of named possibilities. These things are enumerated, but you typically would not do arithmetic on them, except to increment (go to the next) or decrement (go to the preceding). They are distinct (but ordered) possibilities, not quantities.