Homework 1

Due: 17 September 2021

Problem 1.1

Assume that the `Card` class is defined as in Lab 4, and consider the following not-very-useful code:

```
void func (Card a, Card& b, shared_ptr<Card> c)
{
  cout << a.getSuit() << b.getSuit() << c->getSuit() << endl;

  a = Card{9, Card::HEART};
  b = Card{8, Card::SPADE};
  c = make_shared<Card> (1, Card::HEART);

  cout << a.getSuit() << b.getSuit() << c->getSuit() << endl;
}

int main()
{
  Card x = Card{3, Card::CLUB};
  shared_ptr<Card> y = make_shared<Card> (7, Card::DIAMOND);

  func (x, x, y);
}
```

Its output is what you'd probably expect:

```
112
343
```

For this problem, though, draw out two diagrams of what's in memory, and where: one diagram for the state of memory at the time of the first `cout` statement, and another to show the changes as of the second `cout` statement.

To be clear: there are five named variables in this code and all five should appear in both the "before" and the "after" diagrams.

Problem 1.2

Consider a class `RpgPc` that has the task of representing players in a role-playing game, each of which has a name and a series of stats:

```
class RpgPc
{
  private:
    string name;
    unique_ptr<int[]> stats;
...
```

The following is proposed for one of its constructors. Some lines that aren't of current interest are omitted, as is the definition of `average` (which computes the average of a six-element array of integers).

```
RpgPc(string n) : name(n)
{
  unique_ptr<int[]> statsA = make_unique<int[]>(6);
  unique_ptr<int[]> statsB = make_unique<int[]>(6);
  /* ... randomly generate values in both arrays ... */
  double avgA = average(statsA);
  double avgB = average(statsB);
  if (avgA > avgB)
    stats = move(statsA);
  else
    stats = move(statsB);
}
```

a. Draw a diagram of memory at the end of running the constructor; assume for purposes of the diagram that `avgB` was higher than `avgA`.

b. If this code hadn't used managed pointers (`unique_ptr`), but used plain old pointers instead (`int*`, `new`), this would have introduced a memory leak. Explain why, and how to fix it (other than using managed pointers).

Collaboration policy: group work! If you work with other people on this homework, hand in one copy and put all your names on top. There will be a revision cycle for this.

Handing in: on paper in class is fine; otherwise scan/screenshot/photograph into a *single* file and hand in on Canvas. Either way, one handin per group, please.