Discussion #1

assignment operators,copy constructors

Copy Constructors

- `ClassName(const ClassName &classVariable);`
- Default copy constructor copies all values verbatim.
    - This is a problem when it comes to dynamically allocated arrays
    - Both the original and new object's dynamically allocated arrays will point to the same location in memor

Assignment Operators

- `ClassName& operator=(const ClassName &classVariable);`
    - argument is a reference to the item on the right hand side
    - called via `object1 = object2`

```
1  MAX_STUDENTS = 200;
2
3  class Student {
4    public:
5      //...
6    private:
7      string m_name;
8  };
9
10 class School {
11   public:
12     School (const string &name);  // regular constructor
13     School (const School& aSchool); // copy constructor
14                                     // School object passed by reference for
15     string getName() const;
16     void setName(const string &name);
17     string addStudent(const string &name);
18     Student *getStudent(const string &name) const;
19     bool removeStudent(const string &name);
20
21   private:
22     string m_name;
23     Student *m_students;
24     int m_numStudents;
25 };
26
27 School::School(const string &name) {
28   m_name = name;
29   m_numStudents = 0;
30   m_students = new Student[MAX_STUDENTS];
31 }
32
```

```
// COPY CONSTRUCTOR
School::School(const School& aSchool) {
  m_name = aSchool.m_name;
  m_numStudents = aSchool.m_numStudents;
  m_students = new Student[m_numStudents];
  for (int i = 0; i < m_numStudents; i++)
    m_students[i] = aSchool.m_students[i];
}

// ASSIGNMENT OPERATOR OVERLOAD
School& School::operator=(const School& aSchool) {
  if (this != &aSchool)
    return *this;

  if (m_students != nullptr)    // Not explicitly necessary, C++ smart enough to not delete a nullptr
    delete[] m_students;

  m_name = aSchool.m_name;
  m_numStudents = aSchool.m_numStudents;
  m_students = new Student[m_numStudents];
  for (int i = 0; i < m_numStudents; i++)
    m_students[i] = aSchool.m_students[i];
}

School::getName() const {
  return m_name;
}

School::addStudent(const Student &student) {
  m_students[m_numStudents] = &student;
  m_numStudents++;
}
```