Name: 
SSN: 
Score: 
Section:  
(Circle your Section Number below)

<table>
<thead>
<tr>
<th>Section</th>
<th>TA</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>0201</td>
<td>Patrick &amp; Carrie</td>
<td>Monday</td>
<td>7:30 - 8:20pm</td>
<td>ECS 104 A</td>
</tr>
<tr>
<td>0202</td>
<td>Patrick &amp; Carrie</td>
<td>Wednesday</td>
<td>7:30 - 8:20pm</td>
<td>ECS 104 A</td>
</tr>
<tr>
<td>0203</td>
<td>Patrick &amp; Ryan</td>
<td>Monday</td>
<td>2:00 - 2:50pm</td>
<td>ECS 104 A</td>
</tr>
<tr>
<td>0204</td>
<td>Patrick &amp; Ryan</td>
<td>Wed</td>
<td>2:00 - 2:50pm</td>
<td>ECS 104 A</td>
</tr>
<tr>
<td>0205</td>
<td>Patrick &amp; Carrie</td>
<td>Monday</td>
<td>11:00 - 11:50am</td>
<td>ECS 104 A</td>
</tr>
<tr>
<td>0206</td>
<td>Patrick &amp; Carrie</td>
<td>Wed</td>
<td>11:00 - 11:50am</td>
<td>ECS 104 A</td>
</tr>
</tbody>
</table>

Notes:
1. This exam is a closed book, and a closed notes exam.
2. All answers are to be written on the enclosed exam sheets. Scratch sheets are not allowed. If necessary, you can use the back of the exam sheets.
3. You will need to present your Photo ID when handing in the exam. No exceptions.
4. Please hand in your exam with your section number circled. If your section number is not circled, your exam will not be graded.
1) (2 points each) Write True or False in the TRUE/FALSE column:

<table>
<thead>
<tr>
<th></th>
<th>TRUE/FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) argv[1] is always the name of the executable program</td>
<td></td>
</tr>
<tr>
<td>b) Class mutator methods can only be declared as const methods</td>
<td></td>
</tr>
<tr>
<td>c) If a mutator method receives invalid data, then the mutator should terminate the program.</td>
<td></td>
</tr>
<tr>
<td>d) The following vector declaration causes the constructor for the Student class to be invoked: vector &lt;Student&gt; myStudents;</td>
<td></td>
</tr>
</tbody>
</table>
| e) In the method for the post increment operator for the Money class: 
   Money operator++( int dummy); 
   the value of the dummy parameter is zero. |   |
| f) In C++, the stream extraction operator is defined as operator>>. |   |
| g) endl and \n are the same, there is no difference between using endl and \n in a cout statement |   |
| h) A static data member of a class is considered a "global" data member for all instances of the class |   |
| i) Class methods cannot be declared as static methods. |   |
| j) Mr. Raouf mentioned in the lecture that operator overloading is an example of Syntactic Sugar |   |
2. **(6 points)** List 3 issues where vectors are superior to arrays. Give an example for each issue. Use the array declaration `int myArray[10];` for your examples.

Answer:

1. The size of the array must be declared at compile time and cannot be changed (without using dynamic memory allocation).
2. We must use a separate variable to keep track of the number of elements in the array that are actually used.
3. If we use an array index that's out-of-bounds, our program crashes.
4. The name of an array is like a pointer which can be very confusing (e.g. when passed to a function).
5. Elements of an array are not automatically initialized.
6. We can't assign one array to another unless we write code to do so.

3. **(4 points)** Show the output from the following code snippet:

```cpp
vector<int> integers(5);
for (unsigned int i = 0; i < integers.size(); i++)
{
    cout << integers.at(i) << endl;
    integers.push_back(i);
}
```

- What will the value of `i` be after the for loop has completed?
- What will the size of the vector be after the for loop has completed?

**Answer:**

Infinite loop
4. **(4 points)** Rewrite the following code to use call by reference parameters instead of pointers.

   // function prototype
   void add1 (int * pMyInteger);

   //function definition
   void add1 (int * pMyInteger)
   {
       (* pMyInteger)++;
   }

   Answer:

   ```c++
   void add1 (int & MyInteger);
   void add1 (int & MyInteger)
   {
       MyInteger++;
   }
   ```

5. **(6 points)** Write the C++ statements for the following:
   - Create a vector of Bicycle objects called myBicycles
   - Create an instance of a Bicycle class called someBicycle, using the default constructor.
   - Insert the someBicycle object at the end of the myBicycles vector.

   ```c++
   vector <Bicycle> myBicycles;
   Bicycle someBicycle;
   myBicycles.push_back(someBicycle);
   ```
6. **(5 points)** Given that Circle is a class that contains a Point class, and you noticed the following constructor for the Circle class during your visit to the C++ Country Club:

   Circle (Point center, float radius);

   Explain if you should speak out, or remain silent, and why?

   **ANSWER:**

   I would speak out and suggest that passing Point by value is inefficient. I would suggest that the following be used:

   Circle (const Point & center);

7. **(5 points)** Given the following function header for a class accessor called GetDayOfYear, explain the significance of each const listed below:

   const DayOfYear & Vacation::GetDayOfYear ();

   **answer:**

   const DayOfYear & ensures that the method does not alter any of the data members of the class.
8. **(6 points)** What will the following declarations do:

a. `vector <Tractor *> machineShed;`

b. `vector < vector < string> > Words;`

c. `vector <char> WordsInAlphabet(26);`

**Answer:**

a.

b.

c.

9. **(4 points)** Given the following class definition, write the statement to initialize static data member m_turn to 0;

```cpp
class Server
{
    public:
        private:
            static int m_turn;
};
```

**Answer:**

```
int Server::m_turn = 0;
```
10. (5 points) Rewrite the following constructor to use a member initialization list

```cpp
someClass::someClass( string  someName, float someMoney )
{
    m_Name = someName;
    m_Money= someMoney;
}
```

**Answer:**

```cpp
someClass::someClass( string  someName, float someMoney ) :
    m_Name (someName) ,
    m_Money(someMoney)
{
}
```

11. (5 points) Since there is no way to prevent the user of a class from passing invalid parameter values to a constructor, define a mechanism that can indicate to the user of the class if invalid parameters were passed to the constructor.

**ANSWER:**

Create a private Boolean data member that is set to true if the parameter data is correct, and set to false otherwise. Create a public accessor method to return the value of the Boolean data member.

12. (5 points) Given that Toy is a user defined data type, explain each of the following statements:

- Toy myToy();
- Toy myToy;

**ANSWER:**

- Toy myToy(); is a function prototype for a function that returns a Toy object, and is not passed any parameters (void)
- Toy myToy; calls the default constructor for the Toy class to create an object called myToy
13. During one of the lectures where object oriented concepts were first introduced to you, we discussed a conceptual design for a Command Line class. The following is a class definition for CCommandLine class:

```cpp
class CCommandLine
{
    public:
        CCommandLine();
        CCommandLine(int argc, char * argv[]);
        CCommandLine(int argc, const vector <string> & argv);
        string getParameter(unsigned int index);
        bool findParameter(string parameter);
        unsigned int getParameterCount();
        vector <string> getAllParameters();
        const CCommandLine operator+(const string & LHS);
    private:
        int m_argc;
        vector <string> m_argv;
};
```

The class CCommandLine is used to model argc, and argv as defined in main(). The class designer has decided to use 2 private data members:
- int m_argc: used to represent the number of arguments, same as the argc in main
- vector < string > m_argv: used to store the command line parameters, where each parameter is a string, same as the char * argv[] in main.

Given the above class definition, write the code for the following methods:
- **(5 points)** CCommandLine(): default constructor, use a member initialization list where applicable

```cpp
CCommandLine::CCommandLine()
{ }
```

- **(5 points)** CCommandLine(int argc, char * argv[]): constructor, use a member initialization list where applicable

```cpp
CCommandLine::CCommandLine(int argc, char * argv[])
    : m_argc(argc)
    {
        for (int i = 0; i < m_argc; i++)
            m_argv.push_back(argv[i]);
    }
```
• **(5 points)** \( \text{CCommandline}(\text{int argc, const vector & argv}): \) constructor, use a member initialization list where applicable

\[
\text{CCommandline}\text{::CCommandline}(\text{int argc, const vector & argv}) \rightarrow \text{m_argc} = \text{argc} \\
\text{for (int i = 0; i < m_argc; i++;} \\
\text{m_argv.push_back(argv[i]);}
\]

• **(5 points)** bool findParameter(string parameter): a method that returns true if the parameter is found in the parameter list, else returns false

\[
\text{bool CCommandline}\text{::findParameter(string parameter)} \\
\text{unsigned int i = 0; unsigned int size = m_argv.size();} \\
\text{while (i < size)} \\
\text{if (parameter == m_argv[i])} \\
\text{return true; i++;} \\
\text{return false;}
\]

• **(5 points)** const CCommandline\  operator+(const string & LHS): overloaded + operator to add a string to the \text{m_argv}, and to increment \text{m_argc};

\[
\text{const CCommandline CCommandline::operator+(const string & LHS)} \rightarrow \text{m_argc}++; \\
\text{m_argv.push_back(LHS); return CCommandline(m_argc, m_argv);}
\]