Due date: n/a. This is for your practice.
1 GCD greatest common divisor

• This is something you may all have seen before.

• Euclid’s algorithm to calculate the greatest common divisor of two positive integers $a$ and $b$.
  
  1. If $b > a$ swap $a$ and $b$.
  2. Compute the remainder after integer division $c = a \% b$.
  3. If $c == 0$ return $b$.
  4. If $c == 1$ return 1.
  5. Else return gcd(b, c).

• The function signature is as follows.

  ```
  int gcd(int a, int b);
  ```

• Write the function body.
2 Gray Code

- Consider the display of an odometer.
- The digits go 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Then the next is 10.
- **Two rotors have to change, in the tens and units slots.**
- **In a Gray code, only one rotor changes between consecutive numbers.**
- It is easier to illustrate using binary digits.
- For 4 numbers, the ‘odometer’ and Gray code are as follows.

<table>
<thead>
<tr>
<th>odometer</th>
<th>Gray code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

- For 8 numbers, the ‘odometer’ and Gray code are as follows.

<table>
<thead>
<tr>
<th>odometer</th>
<th>Gray code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>101</td>
<td>111</td>
</tr>
<tr>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>111</td>
<td>100</td>
</tr>
</tbody>
</table>

- **Given an input nbits, write a function to generate a Gray code of length \(2^{nbits}\).**
- The function signature is as follows.

```c
void Gray_code(int nbits, vector<string> &gcstr);
```
- The output is a vector of strings of binary bits as shown above for nbits=2 and nbits=3, respectively.
- The calculation does not have to be contained all in one function.
- You may code the function recursively.
- You are permitted to write ‘helper functions’ to perform subsidiary tasks, if needed.
- **However, the calling application will call the above function only, and must receive the correct output of a vector of length \(2^{nbits}\).**
- A Gray code is not unique. All valid solutions are acceptable. But use positive numbers only.
3 Abstract Base Class

• Explain what is an abstract base class.

• Here is a C++ schematic of a class ABC (‘abstract base class’).

```cpp
class ABC
{
    public:
        virtual string name() const;
        virtual void set(int n);
        double sum() const {
            // return sum of array x
        }

    protected:
        int len;
        double *x;
};
```

• Write all additional material required to make ABC an abstract base class.

• Display two different implementations to make ABC an abstract base class.

• The virtual function name() returns the name of the class.

• The virtual function set(int n) allocates x to an array of length n and initializes the array.

• The non-virtual function sum() computes and returns the sum of the array x.

• See next page.
Choose one of your implementations for ABC and write complete code for the following derived classes and make all the class methods work correctly.

class Linear : public ABC
{
    // override name to return "Linear"
    virtual void set(int n) {
        // x[i] = i; // override, initialize x[i] = i
    }
};

class Quadratic : public ABC
{
public:
    // override name to return "Quadratic"
    virtual void set(int n) {
        // x[i] = i*i; // override, initialize x[i] = i*i
    }
};

class Pow_k : public ABC
{
public:
    Pow_k(int ); // non-default constructor, initialize value of k (private member)
    // override name to return "Pow_k"
    virtual void set(int n) {
        // x[i] = pow(i, k); // override, initialize x[i] = pow(i, k)
    }
private:
    int k;
};