1. Consider the following class definition for `Displacement`:

```
using std::ostream;

class Displacement
{   friend Displacement operator*(double, Displacement);
    friend Displacement operator/(Displacement, double);
    friend ostream& operator<<(ostream& ostr, Displacement& x)  {
        ostr << x._m;
    }
  public:
    Displacement(double m=0) : _m(m) { }
    double m() const { return _m; }
    double cm() const { return 100*_m; }
    double km() const { return _m/1000; }
    double in() const { return _m/0.0254; }  // 1 in == 0.0254 m
    double ft() const { return _m/0.0254/12; }
    double mi() const { return _m/0.0254/12/5280; }
    void add_cm(double cm) { _m += cm/100; }
    void add_km(double km) { _m += 1000*km; }
    void add_in(double in) { _m += 0.0254*in; }
    void add_ft(double ft) { _m += 0.0254*12*ft; }
    void add_mi(double mi) { _m += 0.0254*12*5280*mi; }
    void subtract_cm(double cm) { _m -= cm/100; }
    void subtract_km(double km) { _m -= 1000*km; }
    void subtract_in(double in) { _m -= 0.0254*in; }
    void subtract_ft(double ft) { _m -= 0.0254*12*ft; }
    void subtract_mi(double mi) { _m -= 0.0254*12*5280*mi; }
    Displacement& operator=(const Displacement&);
    Displacement& operator*=(const double);
    Displacement& operator/=(const double);
  private:
    double _m;  // meters
};

Displacement operator*(double t, Displacement x)
{ Displacement y(t*x._m);
  return y;
}

Displacement operator/(Displacement x, double t)
{ Displacement y(x._m/t);
  return y;
}

Displacement& Displacement::operator=(const Displacement& x)
{ _m = x._m;
  return *this;
}

Displacement& Displacement::operator*=(const double t)
{ _m *= t;
  return *this;
}

Displacement& Displacement::operator/=(const double t)
{ _m /= t;
  return *this;
}
```

(a) Write out the definition of the constructor and describe the how it works in detail.

(b) Describe how this class overloads the assignment operator.

(c) Write a copy constructor for this class.

(d) What's wrong with using the following code as a copy constructor:

```
Displacement::Displacement( Displacement disp ) :
    _m( disp._m )  {}
```

(e) Write a driver for this class that will test each of the various member functions in a way that's easy to verify, or self-verifying. If might include output such as

```
x1 = 0 meters = 0cm.
x2 = 1 meters = 0.000621371mi.
x3 = 10 meters = 32.8084ft.
x4 = -5 meters = -196.85in.

x5 = 0.043496 miles + 39.3701 inches = 70.0254 meters.
```

etc.

2. Create a class called `PolarCoords` with two private data elements: `Displacement` and `Angle` and provide methods for

(a) overloading the addition operator. Think of the sum as you would the sum of two 2-dimensional spatial vectors.

(b) overloading the insertion operator.

(c) overloading the extraction operator (assume the user enters polar coordinates in the form "(¡r¿,¡theta¿)".

3. Consider the code below defining a `class` named `Queue`.

```
class Queue{
public:
    Queue(int s=100);     // sets the default maximum number at 100
    ~Queue();
    char front() const;
    char back() const;
    bool is_empty() const;
    bool is_full() const;
    void enter(const char);
    char leave();
    void print();
    int size();
private:
    char* _a;      // the queue itself: a dynamic array of char
    int _max;      // the maximum number of elements on the queue
    int _count;    // the number of elements on the queue
};
```

(a) How would you modify this code to make it a template class which could be a Queue of any type of object?

(b) How would you define various functions?

(c) Why is the `front()` function defined as constant?

(d) Why does the `enter()` function take a constant parameter?

(e) Write a definition to implement the `print()` function for the `template class`, Queue, outside the interface.

(f) Write a definition for a `friend` function overloading the insertion operator (`<<`). The code below overloads the insertion operator of the `Date` class, for example.

```
ostream& operator<<(ostream& ostr, const Date& x) {
    switch(x.month()) {
        case 1 : ostr << "January"; break;
        case 2 : ostr << "February"; break;
        // dot dot dot
        case 12 : ostr << "December"; break;
    }
    ostr << " ";
    ostr << x.day() << ", ";
    ostr << x.year();;// << " " << x.era();
    return ostr;
}
```

4. Write code for each of the following:

(a) Declare a pointer `ppc` to a `PolarCoords`.

(b) Create a `PolarCoords pc1` and have `ppc` point to `pc1`.

(c) Create a dynamic array of 3 `PolarCoords` and use pointer arithmetic to intialize, display each of them.

(d) Write code to delete the memory you allocated in part (c).

(e) Suppose that `pdapc` is a pointer to the first of a dynamic array of `PolarCoords`. Write a while loop to add these them all, treating each polar coordinate pair as the terminal point of a vector whose original point is at the origin. Use pointer arithmetic in your loop.

(f) Write code that creates a dangling pointer to a `PolarCoords`.

(g) Write code to dereference a pointer to a `PolarCoord`.

(h) Write code to display the address of a pointer to a `PolarCoord` on the console.

(i) Given the following
`PolarCoords pcArray[5]`
`PolarCoord* pcArrayPtr = pcArray;`
How would you initialize the array to contain the five roots of unity on the unit circle (in polar form) and the use the pointer with pointer arithmetic in a loop to print these to the console?

(j) What type of `const` pointer is this and what permissions does the following expression have?:
`int * const vPtr = &x;`

(k) What type of `const` pointer is this and what permissions does the following expression have?:
`const int *const vPtr = &x;`

(l) What's wrong with the following code?

```
PolarCoord* firstlast(PolarCoord pc[ ], int size) {
  PolarCoord pc1[2];
  pc1[0] = pc[0];
  pc1[1] = pc[1];
  return pc1;
}
```

5. Draw pictures to show the effects of the following statements:

```
string* str1 = new string("Don Knuth");
string str2 = *str1; // str2 is a copy of *str1
string& str3 = *str1; // str3 is a synonym for *str1
string* str4 = &str2; // str4 points to str2
```

```
str4[5] = '?';
str3->erase(3, 2);
str2[1] = 'G';
str1->replace(2, 1,"$=$")
```