CS32 Midterm 2 Study Guide

Recursion

1. Know the rules for writing correct recursive functions (base case, recursive case, etc.).
2. What value is returned by the call `mystery(5);`

```
int mystery(int n)
{
    if (n == 0)
        return 1;
    else
        return 3 * mystery(n - 1);
}
```

3. What value is returned by the call `recur(27);`

```
int recur(int n)
{
    if (n <= 10)
        return n * 2;
    else
        return recur(recur(n / 3));
}
```

4. Write a recursive function that computes x^n. n is a nonnegative integer.

   [Hint: x^n can be defined by the following two equations:

   $x^0 = 1.0$
   $x^n = x * x^{n-1}$ for n >= 1

   ```
   double Power(double x, unsigned int n)
   ```

5. Write a recursive version of

   ```
   double Power(double x, unsigned int n)
   ```

   that works by breaking n down into halves, squaring `Power(x, n/2)`, and multiplying by x again if n was odd.

   For example: $x^{11} = x^5 * x^5 * x$ $x^{10} = x^5 * x^5$

6. Write a recursive function

   ```
   int Product(int m, int n)
   ```

   which gives the product of the integers in the range m:n (m <= n)

7. Write a recursive function

   ```
   int Min(int A[], int n)
   ```

   to find the smallest integer in the integer array A. n is the number of elements in A. HINT: you could define an auxiliary function `Min2(A,k,j)` that finds the smallest integer in A[k:j] and let `Min(A,n) = Min2(A,0,n-1)`

STL/Template

1. Be able to write a simple template function.
2. Be able to create and use a vector, list, stack and queue from the STL. I'll provide for you the functions for each (example: push_back(), etc) so you don't have to memorize them.
3. Know how to iterate through a collection.

Inheritance/Polymorphism

1. What will the following program display?

```
#include <iostream>
using namespace std;

class Person {
    public:
        virtual void eat() {cout << "Yummy" << endl;}
        virtual void speak() {cout << "Hello" << endl;}
        virtual void sleep() {cout << "ZZZZ" << endl;}
        virtual ~Person() {}
};

class Student : public Person {
    public:
        void speak() {cout << "I love school" << endl;}
        void study() {cout << "Studying for Midterm test"
                           << endl;}
```

```
        void getReadyForTest () {
             study();
             sleep();
     }
     virtual ~Student() {}
};

class UCLAStudent : public Student {
    public:
        void speak() {cout << "Go Bruins!" << endl;}
        void sleep() {cout << "ZZZ... CS32 ...ZZZZ"
                           << endl;}
        void getReadyForCS32Test() {
            study();
            eat();
            sleep();
        }
        virtual ~UCLAStudent() {}
};

int main( )
{
    Person* array[3];
    array[0] = new Person();
    array[1] = new Student();
    array[2] = new UCLAStudent();

    for (int i=0; i < 3; i++) {
        array[i]->eat();
        array[i]->speak();
        array[i]->sleep();
    }

    Student * sp = new Student();
    UCLAStudent *uclap = new UCLAStudent();
    sp->getReadyForTest();
    uclap->getReadyForCS32Test();

    return 0;
}
```

2. What is a virtual function?
3. How do you create a pure virtual function?

4. Know what is the difference between overloading a function and redefining a function.
5. Know the difference between static binding and dynamic binding.
6. Should you make a destructor virtual? Why or why not?