Data Structures: 505 22240 / ESOE 2012

Computer Assignment 1

C++ Programming Fundamentals

Due: the week after next by 9:00 p.m.

*Hand in online via **NTU COOL***

Total score: 100

NOTE: please save the source code of each problem in individual cpp files (as specified) WITHOUT the function **main**(). All testing files that have the function **main**() MUST be in separated files.

1. (Save as "CA1Prob1.cpp".) Write a short C++ program that takes two arrays *a* and *b* of length *n* storing integer values, and returns the dot product of *a* and *b*. That is, it computes an array *c* of length *n* such that *c*[i] = *a*[i]·*b*[i], for *i* = 0, …, *n*-1 and returns a scalar representing the dot product. The specification of the function is as follows: int **dotProduct** (int a[], int b[], int n);
   Please consider any possible inputs that causes errors and try to survive them. A simple test with *a* = [-1 0 2 15 7 6 -4 8 21 -13] and *b* = [5 9 -18 16 0 1 -4 18 -2 12] can be used to test your program initially. (20%)

2. (Save as "CA1Prob2.cpp".) Write a C++ function which takes a single integer parameter *p*, and (a) outputs "*p*" if and only if the integer is a prime OR (b) outputs its nearest prime numbers both smaller and larger than *p*: "*ps* and *pl*", where *ps* and *pl* are *p*'s nearest primes smaller and larger than *p*, respectively. To simplify the grading process, please follow exactly the specification as:
   int* **showPrime** (int p);
   In the returned integer array, set the array length equal to two. If *p* is actually a prime, return it in the first element and set the second element 0. If *p* is not a prime, return [*ps pl*] in the array. You may test your program with 5869, 3053 and 104395302. (20%)

3. (Save as "CA1Prob3.cpp".) Write a function that takes two positive **long** integers, *n* and *m*, and returns true if and only if *n* is a multiple of *m*, that is, *n*=*mi* for some integer *i*. (20%)

Please design the function using the following specification:

```
bool isMultiple (long n, long m);
```

4. (Save as "CA1Prob4.cpp".) Write a C++ class, *flower*, that has three member variables of type **string**, **int**, and **double**, which respectively represent the name of the flower, its number of petals, and price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include functions for setting the value of each type, and getting the value of each type. Here is the specification of this class:

```
class flower {
private:
    string name;
    int petal;
    double price;

public:
    // Constructor.
    flower (string n, int pt, double pr);
    // Set the name of the flower.
    void setName (string n);
    // Set the petal number of the flower.
    void setPetal (int pt);
    // Set the price of the flower.
    void setPrice (double pr);
    // Return the name of the flower.
    string getName ();
    // Return the petal number of the flower.
    int getPetal ();
    // Return the price of the flower.
    double getPrice ();
};
```

Please use the following flower data to test your program. (20%)

Flower	**Petal #**	**Price**
Nodding Trillium	3	75.0
Prairie Rose	5	100.0
Bluebell	6	85.0

5. (Save as "CA1Prob5.cpp".) Write a C++ class, *rectangle*, that has four member variables of the same type **double**, which respectively represent the width, length, perimeter, and area. Again, your class must include a constructor method that initializes the width and length of the rectangle and computes the corresponding perimeter and area subsequently. The specification of this class is described in the following: (20%)

```
class rectangle {
private:
    double width;
    double length;
    double perimeter;
    double area;

public:
    // Constructor, automatically compute the perimeter and area once it is OK.
    rectangle (double wd, double lg);
    // Set the dimension and automatically update the perimeter and area.
    void setDimension (double wd, double lg);
    // Set the width and automatically update the perimeter and area.
    void setWidth (double wd);
    // Set the length and automatically update the perimeter and area.
    void setLength (double lg);
    // Return the width of the rectangle.
    double getWidth ();
    // Return the length of the rectangle.
    double getLength ();
    // Return the perimeter of the rectangle.
    double getPerimeter ();
    // Return the area of the rectangle.
    double getArea ();
    // True, if width = length.
    bool isSquare ();
};
```