CS 7A Final Exam Solutions - Spring 2016 - Final Exam 5/24/16

Write responses to questions 1 and 2 on this paper or attach additional sheets, as necessary. For all subsequent problems, use separate paper.
Assume the necessary libraries are included and appropriate namespaces are used.

1. Write the number of the definition on the right next to the term it defines.

   (a) **object** 1

   (b) **variable** 10

   (c) **function** 11

   (d) **declaration** 7

   (e) **parameter** 5

   (f) **byte** 9

   (g) **parser** 8

   (h) **scope** 2

   (i) **overload** 3

   (j) **type** 4

   (k) **reference** 6

   (1) Some memory that holds a value of a given type.

   (2) The region of program text (source code) in which a name can be referred to.

   (3) Defining two functions or operators with the same name but different argument (operand) types.

   (4) A set of possible values and a set of operations (for an object).

   (5) A declaration of an explicit input to a function through which a function can access the arguments passed by name.

   (6) A value describing the location of a typed value in memory.

   (7) The specification of a name with its type in a program.

   (8) A program that reads a stream of tokens according to a grammar.

   (9) The basic unit of addressing in most computers.

   (10) A named object of a given type; contains a value unless uninitialized.

   (11) A named unit of code that can be invoked (called) from different parts of a program; a logical unit of computation.

2. Write a C++ program to implement the pseudocode:

```
Set total to zero
Set temperature counter to one
While temperature counter is less than or equal to ten
    Input the next temperature
    Add the temperature into the total
Set the average temperature to the total divided by ten
Print the average temperature.
```

```
1 int main()
  {   double total{}, temp{}, avgTemp{}, tempCtr{1};
3     while(tempCtr<11)
      {   cin >> temp;
5         total += temp;
      }
7     avgTemp = total/tempCtr;
      cout << "\nThe␣average␣temperature␣is␣" << avgTemp;
9 }
```

3. Consider the following program:

```
1  void getCoordinates(int&,int&,int&,int&);

3  int main() {
       int x1 = 1, y1 = 2, x2 = 3, y2 = 4;
5      getCoordinates(x1, y1, x2, y2)
       //slope(/*parameters*/); //uncomment: print slope; specify input
7  }
   void getCoordinates(int& x1, int& y1, int& x2, int& y2)
9  {   cout << "\nEnter␣coordinates␣x1,␣y1,␣x2,␣y2,␣in␣that␣order:␣";
       for(int i = 0; i < 4; i++)
11     {   switch (i)  {
           case 0:
13             cin >> x1; break;
           case 1:
15             cin >> y1; break;
           case 2:
17             cin >> x2; break;
           case 3:
19             cin >> y2; break;
           }
21     }
   }
```

Remember to write your answers to these on separate paper.

(a) Why is the `&` symbol used on line 8? Explain what that means in that context and why it's needed.
ANS: The reference operator creates an alias (another name) for the variable passed, i.e. the second variable `x1` has the same address (perhaps, confusingly, with the same name, `x1`). It doesn't contain address itself, it just references the same portion of memory as the variable it's initialized from. This is needed here so that the values passed by reference are actually themselves changed, rather than just copies of them, and when control is returned to `main()`, the changes to the passed variables remain changed.

(b) What single `cin` statement could replace the `for` loop and `switch` statement in `getCoordinates()`?
`cin >> x1 >> y1 >> x2 >> y2;`

(c) Write code to define a `class Point` with two member variables `x` and `y` and a constructor, `Point(x,y)` to instantiate such a point. This allows you to create a `vector<Point>` which `getCoordinate()` can take as a reference parameter.

```
1  class Point {
   public:
3          double x;
           double y;
5          Point(double xx, double yy) : x(xx), y(yy) {}
   };
```

(d) Rewrite `main()` and `getCoordinate()` to use a `vector<Point>` to get a `vector` of 100 `Point`s available for further processing in `main()`.

```
   #include "std_lib_facilities.h"
2
   const int N{3};
```

```
4
class Point {
6 public:
      double x;
8     double y;
      Point() {}  // default constructor
10    Point(double xx, double yy) : x(xx), y(yy) {}
};
12
///prototype
14 void getCoordinate(vector<Point>&);

16 int main()
{   vector<Point> vp(N);
18    getCoordinate(vp);
}
20
void getCoordinate(vector<Point>& vp)
22 {   double x1{}, y1{};
        for(int i = 0; i < vp.size(); ++i)
24    {   cin >> x1 >> y1;
        vp[i] = Point(x1,y1);
26    }
}
```

(e) Write the function `slope(p1,p2)` to take two `Point` variables and return the slope of the line between them as a floating point approximation, or print an error message if the slope is undefined. For instance, if `Point p1(1,2)` and `Point p2(3,5)` are passed to the `slope()` function, then it will print to the console

```
The slope of the line from (1,2) to (3,5) is 1.5
```

Whereas, if `Point p1(1,2)` and `Point p2(1,5)` are passed, it will print

```
The slope of the line from (1,2) to (1,5) is undefined.
```

```
1
double slope(Point p1, Point p2) {
3     if(p1.x==p2.x)
      {   cout << "\nThe␣slope␣of␣the␣line␣from␣(" << p1.x
5               << "," << p1.y << ")␣to␣("
                << p2.x << "," << p2.y << ")␣is␣undefined.";
7         return 999999999.;
      }
9     else
      {   double slope = (p2.y-p1.y)/(p2.x-p1.x);
11        cout << "\nThe␣slope␣of␣the␣line␣from␣(" << p1.x
                << "," << p1.y << ")␣to␣("
13              << p2.x << "," << p2.y << ")␣is␣" << slope;
          return slope;
15    }
}
```

(f) Write the function `allSlopes()` that takes a `vector<Point>` and returns a count of the number of vertical lines and computes a `vector<double>` that contains all possible slopes between any two `Points` in the input `vector<Point>`.

ANS: Here is a complete program showing `allSlopes()` together with the context in which it may be used:

```
#include "std_lib_facilities.h"
const int N{5};

class Point {
public:
    double x, y;
    Point() {}
    Point(double xx, double yy) : x(xx), y(yy) {}
};

///prototypes
void getCoordinate(vector<Point>&);
double slope(Point, Point);
int allSlopes(const vector<Point>, vector<double>&);

int main()
{   vector<Point> vp(N);
    cout << "\nEnter " << N << " points: " << endl;
    getCoordinate(vp);
    vector<double> slopes;
    cout << "\nThere are " << allSlopes(vp,slopes) << " vertical lines in this
    cout << "\nThe slopes are " << endl;
    for(double d : slopes ) cout << d << " ";
}

void getCoordinate(vector<Point>& vp)
{   double x1{}, y1{};
        for(int i = 0; i < vp.size(); ++i)
    {   cin >> x1 >> y1;
        vp[i] = Point(x1,y1);
    }
}

double slope(Point p1, Point p2) {
    if(p1.x==p2.x)
    {   cout << "\nThe slope of the line from ("
             << p1.x << "," << p1.y << ") to ("
             << p2.x << "," << p2.y << ") is undefined.";
        return 999999999.;
    }
    else
    {   double slope = (p2.y-p1.y)/(p2.x-p1.x);
        cout << "\nThe slope of the line from ("
             << p1.x << "," << p1.y << ") to ("
             << p2.x << "," << p2.y << ") is " << slope;
        return slope;
    }
}

int allSlopes(const vector<Point> vp, vector<double>& slopes) {
    int vertCount{};
```

```
    double deltaX{};
    cout << vp.size();
    for(int i = 0; i < vp.size(); ++i)
    {   for(int j = i+1; j < vp.size(); ++j)
        {   deltaX = vp[i].x-vp[j].x;
            cout << "\ndeltaX␣=␣" << deltaX << endl;
            if(deltaX==0) ++vertCount;
            else slopes.push_back((vp[i].y-vp[j].y)/deltaX);
        }
    }
    return vertCount;
}
```

4. In the code below, suppose that `D=864` and `primes[1]=2`, etc., so that `primes[i]` is the ith prime.

```
int findGreatestSquF(int D) {
    int i = 1, returnVal = 1;
    while(D/primes[i] > sqrt(D))
    {   while(D%(primes[i]*primes[i])==0)
        {   returnVal *= primes[i];
            D /= primes[i]*primes[i];
        }
        ++i;
    }
    return returnVal;
}
```

(a) Complete that table below which threads the values of various quantities as the algorithm is executed:

D	i	primes[i]	returnVal	D/primes[i]	$\sqrt{D}$	D%(primes[i]*primes[i])
864	1	2	1	432	29.4	0
216	1	2	2	108	14.7	0
54	1	2	4	27	7.3	2
54	2	3	4	18	7.3	0
6	2	3	12	2	2.4	6

(b) What value will `findGreatestSquF()` return for `D=864`?
ANS: 12, the largest number whose square is a divisor of 864.

5. Consider the following function.

```
/// count the relative frequency of the alphabetic characters in a file
void letterFreq(vector<double>& charFreqs) {
    ifstream ifs("file.txt");
    char ch;
    double counter{};
    while(ifs>>ch) {
        if(!isalpha(ch)) continue;
        ++counter;
        ch = toupper(ch);
        ++charFreqs[int(ch)-65];
    }
    for(int i = 0; i < charFreqs.size(); ++i)
        charFreqs[i] /= counter;
}
```

(a) Describe the input parameter for this function. What is it? How must it be initialized in order to work properly here?
ANS: The input, `charFreqs`, is a `vector<double>` . It should be initialized like so

```
vector<double> charFrequs(26);
```

to assure that it has enough space to hold frequencies for each of the 26 letters of the alphabet.

(b) Describe what happens in line 3.
ANS: In line 3 an input stream is created to read from a file named `file.txt`.

(c) How will the `while` loop terminate here?
ANS: When the command `ifs>>ch` reaches the end of the file, the `eof()` flag will be set to true and the `istream` will go into `fail` mode.

(d) Describe what happens in line 7.
ANS: If the character `ch` read from the file is not an alphabetic file, skip the rest of the body of the while loop and read the next character.

(e) Describe what happens in line 10.
ANS: At this point we know that `ch` is an upper case character whose ASCII value is between 65 and 90 so that `int(ch)-65` is a number between 0 and 25, inclusive. To increase the frequency count for this character, `charFreqs[int(ch)-65]` is incremented.

(f) What will `vector<double> charFreqs` contain after return to the calling function if `file.txt` contains the single line

```
Pack my box with five dozen liquor jugs.
```

According to Wikipedia, this sentence is used on NASA's Space Shuttle.
ANS: The counts are {1,1,1,1,2,1,1,1,3,1,1,1,1,1,3,1,1,1,1,1,2,1,1,1,1,1} and there are 32 characters, so the frequencies are
$\{\frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{3}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{3}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}, \frac{1}{32}\}$