1. **Fill in the code to make this program work.**

```
#include<iostream>
#include<_______>
#include<cstdlib>
#include<string>
using namespace std;

void inOut (ofstream&); //function prototype

int main()
{
   string ______ ="list.dat"; // here is the file you are
                              // working with
   ofstream outFile;
   ______.open(fname.c_str());
   if (outFile.______()) // check for a successful open
   {
      cout << "The output file was not successfully opened."
           << endl;
      exit(1);
   }
   inOut(______); // call the function
   ______ 0;
}

______ inOut (ofstream& fileOut)
{
   const ____ NUMLINES = 5; // number of lines of text
   string line;
   int ______;
   cout << "Please enter five lines of text: " << endl;
   for (count = 0; count < NUMLINES; count++)
   {
      getline (___,line);
      fileOut << ______ << endl;
   }
   cout << "The file has been successfully written." << endl;
   return;
}
```

2. **Complete the code for the following noDuplicates function that returns a bool result. Its purpose is to check whether its first argument, an integer array of numbers, contains any duplicates. It is also passed the size of the array to allow for loop processing. If the array has any duplicates the function should return false otherwise it returns true. The algorithm for the program is to compare each number in the array to every other number in the array. If a match is found return false. After all comparing is done and no matches have been found, return true.**

```
//function prototype
bool noDuplicates(int [ ], int);

//function definition
bool noDuplicates(____ numbers_____, ____ size)
{
   for (int i=0; i < _______; i++)
   {
      for (int j = 0; j < _______; j++)
      {
         if (j !=____ )
         {
            if( __________[___] == __________[___])
            {
               return _________;
            }
         }
      }
   }
   return _________;
}
```

3. **Write the code to define a function, CalculateMean, which takes as its first argument (an input argument) an array of floating point values, as its second argument (an input argument) an integer value that contains the number of data items in the array. The function should be a value returning function that calculates and returns the mean (average) of the data items in the array. The array should be able to contain values accurate to 14 significant digits. The return value should be able to contain a value accurate to 14 significant digits.**

4. **Consider the following code that will assign a letter grade of 'A', 'B', 'C', 'D', or 'F' depending on a student's test score. Will it work correctly? If not, for what values of score will it work correctly and how would you fix it?**

```
if (score >= 90)
{
   grade = 'A';
```

```
}
if (score >= 80)
{
   grade = 'B';
}
if (score >= 70)
{
   grade = 'C';
}
if(score >= 60)
{
   grade = 'D';
}
else
{
   grade = 'F';
}
```

5. **Given the following integer array:**

   23, 5, 16, 33, 7

   Draw out the steps for how bubble sort would sort the array so that the result is

   5, 7, 16, 23, 33

6. **Write the code that would be in the file Point2D.h for the class Point2D. The class should model a two-dimensional point (x, y), where x is the horizontal component of the point and y is the vertical component of the point. The x and y components of the point are the basic data for the class.**
   - The class should contain a default constructor that sets the x and y components to zero.
   - The class should contain a constructor that takes values to assign to the x and y components.
   - The class should contain accessor and mutator functions for the x component of the point. (NOTE: Remember the accessor functions are the “get” functions, while the mutator functions are the “set” functions.)
   - The class should contain accessor and mutator functions for the y component of the point.
   - The class should contain a member function, Scale, that will scale x and y components of the point by the scale value passed to the function Scale.