Assignment 4

Function prototypes:
An additional point about function prototypes that we did not emphasize in lecture is that you do not need a separate function prototype if you define the entire function above the point where it is called. If you define square before main, for instance, you do not need a prototype for square.
A note on global variables: We discussed in lecture how variables can be declared at *file scope* – if a variable is declared outside of any function, it can be used anywhere in the file. For anything besides global constants, this is usually a bad idea – if you need to access the same variable from multiple functions, most often you simply want to pass the variable around as an argument between the functions. Avoid global variables when you can.

Part 1 – Fix the Function:

1. Identify the errors in the following programs, and explain how you would correct them to make them do what they were apparently meant to do.
a.

```
#include <iostream>
int main()
{
  cout << square(35);return 0;
}
int square(int number) { return number * number; }
```

b.

```
#include <iostream>
int square();
int main()
{
 int number = 35;
 cout << square(number);
 return 0;
}
int square()
 { return number * number; }
```

(Give two ways to make this program work. Indicate which is preferable and why.)

c.

```
#include <iostream>
void square(int number);
int main()
{
  int numberToSquare = 35;
  square(numberToSquare);
  cout << numberToSquare; // Should print 35^2
  return 0;
}
void square(int number)
  { number = number * number; }
```

(For this problem, do not change the return type of the square function.)

d.

```
#include <iostream>
int sumOfPositives(int a, int b);
int main()
{
  cout << sumOfPositives(35, 25, 3);
  return 0;
}
int sumOfPositives(int a, int b)
  {
    int sum = 0;
    if(a > 0)
      sum += a;
    if(b > 0)
      sum += b;
}
```

(Two mistakes, one of which you may fix in either of two ways.)

2. What do the following program fragments do? (Try to answer without compiling and running them.) Briefly explain your results.

a.

```
char *hello = "Hello world!";
void printHello(char *hello)
 {cout << hello;}
int main()
{  printHello("This is not the hello message!");}
```

b.

```
int mystery(int &number)
{ std::cout << number;
  return ++number; }
int main()
{
   for(int i = 0; i < 10; i++)
   {
     cout << mystery(i);
   }
}
```

c.

```
int mystery(int number)
 { cout << number;
   return ++number; }
int main()
{
  for(int i = 0; i < 10; i++)
  {
    cout << mystery(i);
  }
}
```

Part 1: Solution:

1. a. A function prototype for square should be added before main (or the whole function should be moved earlier).
b. Either number should be added as a parameter to square, or number should be made a global variable and removed from the function call to square. The former method is preferable, since the latter entails creating global variables unnecessarily.
c. Change the square prototype and definition to take an int&.
d. sumOfPositives should return sum. Also, the call to it has too many arguments, which can be corrected either by adding an argument to the prototype and definition or by deleting one of the arguments.

2. a. It prints "This is not the hello message!".
b. It prints the numbers from 0 to 9 (the print statements that actually print these numbers alternate between main and the function).

c. It prints the numbers from 0 to 10, repeating each of the numbers in between twice. There are two print statements: the one in mystery gets executed for a given number, and then, since the parameter was passed by reference, the original counter is not updated by mystery, but only by the for loop, so the same number is printed again by the cout statement in main.

Part 2:

Write a C++ program to input elements of one dimensional array with size n and define functions to do the following:

1- Print array elements.
2- Calculate the sum of array elements.
3- Calculate the mean.
4- Calculate the variance.
5- Calculate the standard deviation.
6- Calculate the range.
7- Find the max element.
8- Find the minimum element.
9- Sort array elements.
10- Calculate the median.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void printout(float A[],int size);
float sum(float A[],int size);
float mean(float A[],int size);
float variance(float A[],int size);
float stddev(float A[],int size);
float coeffvar(float A[],int size);
float max(float A[],int size);
float min(float A[],int size);
float range(float A[],int size);
void sort(float A[],int size);
float median(float A[],int size);
```

```
void main()
{
  const int n=10;
  float M[n];
  int i;

  for (i=0;i<n;i++)
   {
    printf("\n Matrix[%d] = ",i+1);
    scanf("%f",&M[i]);
   }

      printout(M,n);
       printf("\n the sum of the matrix elements = %6.3f", sum(M,n));
       printf("\n the mean of the matrix elements = %6.3f", mean(M,n));
       printf("\n the variance of the matrix elements = %6.3f", variance(M,n));
       printf("\n the standard deviation of the matrix elements = %6.3f", stddev(M,n));
       printf("\n the coefficient of variation of the matrix elements = %6.2f %", coeffvar(M,n));
       printf("\n the maximum element = %6.2f", max(M,n));
       printf("\n the minimum element = %6.2f", min(M,n));
       printf("\n the range of the matrix elements = %6.3f", range(M,n));
       sort(M,n);
       printout(M,n);
       printf("\n the median of the matrix elements = %6.3f", median(M,n));

  getch();
}

void printout(float A[],int size)
{
  int i;
  for(i=0;i<size;i++)
    printf("\n %6.2f",A[i]);
  return;
}



float sum(float A[],int size)
{
  int i;
  float s=0.0;
  for(i=0;i<size;i++)
     s+= A[i];
  return s;
}
```

```
float mean(float A[],int size)
{
  float m;
  m = sum(A,size)/size;
  return m;
}

float variance(float A[],int size)
{
  int i;
  float s1=0.0,s2=0.0,m1,m2,v;
  for(i=0;i<size;i++)
    {
     s1+= A[i];
     s2+= pow(A[i],2);
    }
   m1=s1/size;
   m2=s2/size;
   v = m2 - m1*m1;
  return v;
}
float stddev(float A[],int size)
 {
   float st;
   st = sqrt(variance(A,size));
   return st;
 }
float coeffvar(float A[],int size)
{
  float cov;
  cov = (stddev(A,size)/mean(A,size))*100;
  return cov;
}

float max(float A[],int size)
{
  int i;
  float m;
  m = A[0];
  i=1;
  while (i<size)
  {
    if (m< A[i])  m=A[i];
    i++;
  }
  return m;
}
```

```
float min(float A[],int size)
{
  int i;
  float m;
  m = A[0];
  i=1;
  while (i<size)
  {
    if (m> A[i])  m=A[i];
    i++;
  }
  return m;
}

float range(float A[],int size)
{
 float r;
 r = max(A,size) - min (A,size);
 return r;
}

void sort(float A[],int size)
{
 int i,j;
 float temp;
 for(i=0;i<size;i++)
   for(j=i+1;j<size;j++)
    if (A[i]>A[j])
     {
       temp = A[i];
       A[i] = A[j];
       A[j] = temp;
     }
 return;
}

float median(float A[],int size)
{
 float m;
 if (size%2==0)
    m = (A[size/2-1]+A[size/2])/2.0;
 else
    m = A[(size-1)/2];
 return m;
}
```