Fall 2015 Biostat 615 Homework #1 (Total 40 pts)

Due by Wednesday September 30th, 2015 noon as a compressed file "hw1.tar.gz" containing the required C++ source code files for the following four problems. Please read the instruction carefully before you start to work on your homework.

Instruction

- The submission procedure is the same as the one for homework #0.
- You are encouraged to submit the homework much earlier than the above deadline to receive additional grading and feedback before your final submission. You are free to update your submission at any time before the deadline. There will be five times of additional grading before your final submission: September 21st Noon, September 23rd Noon, September 25th Noon, September 27th Noon and September 29th Noon. The final grading will be executed exactly at 12:10pm, Wednesday September 30th, 2015.
- All your programs should get input from `argv`. In the following examples the first line is the command line and the second line is the expected output of your program.
- Make your algorithm as efficient as you can, as some of the test cases may be computationally challenging and your program will be terminated if it does not finish after running for 10 seconds and you will lose the points for those test cases. However, you should always turn in a program even if it is not very efficient so that you can at least get partial points.
- All your programs (e.g., if it is named as `program.cpp`) will be compiled and tested on the server `scs.itd.umich.edu` using the following command:

```
g++ -O -o program program.cpp
```

  where "program" will be replaced by the corresponding program name for each problem.

- For this homework, you are NOT allowed to use your own header files in your submission.

Problem 1 - Calculate Standard Deviation (10 pts)

Write a C++ program `standardDeviation` that returns the standard deviation of all input argument values. You may assume that all input arguments are numeric values and use `atof()` function to convert the argument into a `double` type variable. You may use function `sqrt` by including header `#include<cmath>`. Note that your C++ source code file must be named as `standardDeviation.cpp`. The output value of the standard deviation should have an 8 digit precision. No error handling for malformed argument is needed. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw1$./standardDeviation 1 2 3 4
1.2909944
user@host:~/Private/biostat615/hw1$./standardDeviation 1.01010 2.30034 2.2131
0.72105699
```

Problem 2 - Fit A Simple Linear Regression (10 pts)

Write a C++ program `simpleLinearRegression` to fit the following simple linear regression model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i,$$

where $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$, predictor $X_i = i$ and outcome Y_i is determined by input argument of program, for $i = 1, \ldots, n$. The positive integer n is the sample size which is determined by the number of Y_i's that are entered into the argument list of the program. The program computes the least squares estimates of β_0 and β_1 which are given by

$$\widehat{\beta}_0 = \overline{Y} - \widehat{\beta}_1 \overline{X} \text{ and } \widehat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2},$$

where $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$. You may assume that all input arguments are numeric values and use `atof()` function to convert the argument into a `double` type variable. Note that your C++ source code file must be named as `simpleLinearRegression.cpp`. The input arguments of the program are a number of Y_i's and the output prints $\widehat{\beta}_0$ and $\widehat{\beta}_1$ in one line but separated by a white space. Both of them should have an 8 digit precision. No error handling for malformed argument is needed. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw1$./simpleLinearRegression 1 2 3 4
0 1
user@host:~/Private/biostat615/hw1$./simpleLinearRegression 1.93432121 2.13
1.7386424 0.19567879
```

Problem 3 - Find Binary Representation of Floating Point Number (10 pts)

Write a C++ program `floatBinaryRep` to find the binary representation of a single precision floating point number in C++ based on the definition by the IEEE 754 standard. The program also returns the actual decimal values of the binary representation. In the program, you may use `double` type variables and function `pow` with including header `#include<cmath>`, while you are NOT allowed to use function `reinterpret_cast` (For those of you who are not familiar with this function, `reinterpret_cast` can convert between types by reinterpreting the underlying bit pattern. I would like you to avoid to use this function in that this homework problem is designed for the C++ programming practice and helps you to have a deeper understanding of the IEEE 754 definition). Note that your C++ source code file must be named as `floatBinaryRep.cpp`. The input argument value is a real number to be converted. The output of the program has two lines: the first line prints the binary representation and the second line prints the actual decimal value of the binary representation. No error handling for malformed argument is needed. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw1$./floatBinaryRep 2015.915
01000100111110111111110101001000
2015.9150390625
user@host:~/Private/biostat615/hw1$./floatBinaryRep -9.25
11000001000101000000000000000000
-9.25
```

Problem 4 - Compute Predictive Probabilities (10 pts)

Write a C++ program `predProb` to compute the predictive probabilty of a fitted five-categray logistic regression model for the classification problem:

$$\widehat{p}_k := \widehat{\Pr}(Y = k \mid X_1, \ldots X_p) \propto \exp\left(\widehat{\beta}_{k,0} + \sum_{j=1}^{p} \widehat{\beta}_{k,j} X_j\right), \quad \text{for } k = 1, \ldots, 5,$$

where the estimated coefficient $\widehat{\beta}_{k,j} = 2^{|k-j|}$ for $j = 1, \ldots, p$ and $\widehat{\beta}_{k,0} = 2^{-k}$ for $k = 1, \ldots, 5$. The predictor $X_j \in [0, 1]$ for $j = 1, \ldots, p$ is determined by the input argument of program, for $j = 1, \ldots, p$. The positive integer p is the number of predictors in the model. It is determined by the number of X_j's that are entered into the argument list of the program. The notation "$a_k \propto b_k$" means that a_k/b_k be a constant for $k = 1, \ldots, 5$. You may assume that all input arguments are numeric values and use `atof()` function to convert the argument into a `double` type variable. You may use functions `exp`, `pow` and `abs` by including the header `#include<cmath>`. Note that your C++ source code file must be named as `predProb.cpp`. The input arguments of the program are a number of X_j's and the output prints $\widehat{p}_k$, for $k = 1, \ldots, 5$, in five lines. All of them should have an 8 digit precision. No error handling for malformed argument is needed. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw1$./predProb 0 0.1
0.23143717
0.16309102
0.15906429
0.18251063
0.26389689
user@host:~/Private/biostat615/hw1$./predProb 0 0.1 0 0.5
0.82632972
0.078806373
0.028275448
0.019677854
0.046910609
```