MAT 362 Spring 2007 1. Programming assigment: data analysis Name:

Instructor: Nándor Sieben
e-mail: nandor.sieben@nau.edu
http://jan.ucc.nau.edu/∼ns46/362/362.html
Office: AMB 175
Hours: MW 10:10-12:10, F 10:10-11:10

Write a program that reads lines from standard input and writes a few statistical values to standard output. Run your program with the first sample input file and verify that your output is the same as the sample output file. Run your program with the second input file.
Follow the example source code `swap.cc` on the website to write your source file. Use consistent indentation and write plenty of comments in your code. Use meaningful variable names.

Turn in:

- This problem sheet with your name.
- A summary sheet explaining what you did, how you approached the problem, what was accomplished, what was not accomplished, etc.
- The output file gotten from the second input file.

Website:

- Create a directory called `1dataanal` on your web site and make all your input, output and source files available in this directory. Write the url for the website on this problem sheet.

Input:

An arbitrary number of lines. Each line contains three doubles representing the length, width and height of a brick.

Output:

Write each of these values in a separate line.
- The number of bricks.
- The smallest surface area of a brick.
- The largest volume of a brick.
- The average volume of the bricks.
- The number of bricks having the largest volume up to absolute tolerance 0.01.

Sample input:

```
10 2 1
20 1 1.0001
1 3.4 2
```

Sample output:

```
3
24.4
20.002
15.6007
2
```

Hints:

- Write plenty of comments. Write a coment after every variable declaration.
- Use `indent -br dataanal.cc` to indent your code.
- To compute the last output you need to store the volumes. Since there is no limit on the number bricks you cannot allocate statically an array. Use the `vector<double>` data type to store the volumes. You need to include `vector` for this. Take a look at `vector.cc` for an example.
- Take a look at `swap.cc` to see how to know how many lines are in the input.
- Use the `fabs` function to decide if two numbers are the same up to tolerance. You need to include `math.h` for this.
- Never hardwire any constants into your code unless it's absolutely unavoidable. For example use `const double tol=0.01`.