CS 251 Spring 2009

Due: Thursday Jan 22nd, 11:59PM.

A1—Basic C++

In a nutshell: extend the functionality of the code developed in class; start from the A1.zip code archive provided in the same directory as this document.

Specifics:

1. For our matrix class **M2x2**, implement:
    a. An **operator<<** that displays the values stored in the matrix. See sample output for details on required output formatting. Warning: pay very close attention to the required output format as an error here can be costly.
    b. A **Transpose** method that returns a new matrix obtained by transposing the matrix on which it was called.
    c. A **Column(int i)** method that returns a new V2 containing column i. The first column is column 0.
    d. A **M2x2(float theta)** constructor that builds a rotation matrix. theta is in degrees.
2. Implement a new class **Seg2D** that models 2-D line segments (NOT lines). The class should store the two endpoints. The class should provide the following functionality:
    a. A constructor **Seg2D(V2 p0, V2 p1)** that builds a segment from two endpoints. p0[0] is the x coordinate, p0[1] is the y coordinate, etc.
    b. A method **Length()** that returns the length of the segment
    c. A method **bool Intersect(Seg2D s1, V2& intPoint)** that computes the intersection between the current segment and s1. If there is an intersection, the method should return true and should set **intPoint** to the intersection point. If there is no intersection, the method should return false and should not modify **intPoint**. Assume that s1 and s2 overlap in at most 1 point. Assume infinite precision arithmetic. Remember that the intersection of line segments is different than the intersection of lines.
3. OPTIONAL Extra credit 2%:
    a. Implement a method **bool Circle::Tangent(V2 pt, V2 tgtPoints[2])** that returns true if there is a tangent from point **pt** to the circle and false otherwise. If there are tangents,

the function should set tgPoints to the tangent points. Ignore the case when the point is on the circle.

b. Your code must work perfectly to receive any extra credit. No partial extra credit.

c. Turnin a blank file named "extracredit" for your extra credit to be graded (do NOT create this file if you did not attempt the extra credit). Create this file in the A1 directory with the command

```
touch extracredit
```

4. Sample output: for the given main.cpp, the provided file "sampleoutput.txt" contains the expected output. In particular, use this output to determine the expected output for the << operator for M2x2. When your program is correctly working, the unix command "diff" should report no difference (no output from diff) between your output from "main" saved into a file "myoutput.txt" and sampleoutput.txt. Test this by:

```
./main > myoutput.txt

diff myoutput.txt sampleoutput.txt
```

a. of course once you get this to work you should devise additional test cases to ensure your program truly works

5. Requirements:

1. You MUST use the base code given along with the assignment in the archive A1.zip
2. Do not change the names of any of the files given.
3. Do not include code in files other than those given.
4. Do not change the signatures of any of the functions given.
5. You may add functions and data as you need.
6. Use the Makefile given. Do not change it. Your program should compile and produce the executable "main" by simply typing "make". This has already been setup for you. Do not turn in Makefile, we will use our own that is the same as the one given to you.
7. Your code MUST compile on lore.cs.purdue.edu for it to be graded.
8. Do not include any code you want graded in main.cpp. It will be replaced during grading. Do not turnin main.cpp

6. Turn in instructions

a. any turnin instructions listed here override any “general turnin instructions” listed on the course webpage.

b. all of your code (circle.cpp, circle.h, m2x2.cpp m2x2.h, seg2d.cpp, seg2d.h, v2.cpp, v2.h, extracredit(OPTIONAL)) MUST be in the directory A1 (the same directory that the assignment was distributed in). Other files in the directory will not be graded.

c. in the directory above A1, run the command on lore.cs.purdue.edu

```
turnin -c cs251 -p A1 A1
```

d. optionally run the following command to verify what files were submitted

```
turnin -c cs251 -p A1 -v
```

e. Warning: turnin will be automatically disabled at 01/23/2009 at 12:00 AM EST and you will be unable to turnin your assignment after this time.

f. Warning: turning in multiple times is ok but only the last one will be graded as each turnin permanently overwrites the previous one.

g. Warning: failure to turnin exactly according to the instructions given above will result in your A1 receiving a grade of 0.

7. Grading

a. grading will be primarily through testing the output of each of the required functions against the desired output via a script. Thus it is important that your output format match the expected output format.

Voicu Popescu, spring 2009