Tutorial Instructions

Lectures 3 and 4

1 Hello world

Write a program that prints "Hello, world!", and save it as `helloWorld.cpp`. Compile it and run it from a terminal.

```
g++ -o helloWorld helloWorld.cpp
./helloWorld
```

2 Containers

The file `maxVal.cpp` contains functions that are supposed to find the maximum value in various containers. Implement them, then run the test program to see if you did it right. Recall that a `map` stores key-value pairs. The `map` version should find the maximum key and return the corresponding value.

```
g++ -o test test.cpp maxVal.cpp
./test
```

3 Palindromes

The file `palindrome.h` declares functions for determining whether or not a given string is a palindrome. However, as you will see in `palindrome.cpp`, all the important ones have not been implemented. Your task is to write the functions so that the program in `test.cpp` completes successfully. A palindrome, as you know, is a string that reads the same forward and backward. The word 'mom' is a palindrome, and the phrase 'Never odd or even' is a palindrome if case and spaces are ignored. The program can be compiled and run as follows.

```
g++ -o test test.cpp palindrome.cpp
./test
```

4 Pointers

The file `pointerTest.cpp` contains a bunch of functions that manipulate pointers. Your task is to edit the *body* (not the signature) of these functions so that their output satisfies the assertions performed in the `main` function.

5 Debugging time

The file `debugThis.cpp` contains a program that is supposed to invert a function, but it crashes when run. Your task is to fix it. Once you have fixed the program so that it runs as intended, you will see that it's actually quite slow. Make it run faster! Alternatively, if you don't know how to write the code, think about how you could make the program run faster and discuss your answer with the teacher.

The file `debugThis2.cpp` contains a simple program with a few print statements. Study the code and try to guess what will be printed, then run the program and see if you were right. Some of the results will probably be unexpected. Your task is not to fix the program, but to understand *why* it printed what it did. Fiddle with the code and try to figure it out, then discuss your answer with the teacher.

6 Lists of integers

The files `num1.dat` and `num2.dat` contain lists of integers. We want to know which numbers are present in `num1.dat` but not in `num2.dat` and vice versa. Write a program that performs this task, then compile and run it (you should know how by now). Of course, we will want to compare other files in the future, so the names `num1.dat` and `num2.dat` should not be hard-coded. Let the user provide them from the command line.

Answer: The numbers unique to `num1.dat` are 90721, 2080770 and 2436800 while the numbers unique to `num2.dat` are 154321, 518479 and 2469334.

7 Significant points

We have built a detector to monitor suspicious alien activity. The detector, which is pointed at space, counts the number of detected subspace transmissions during a certain period of time and writes this data to a file called `detReadout.dat`. Each row of the file has three numbers. The format is <*time*> <*counts*> <*flag*>. Table **??** explains the meaning of each field.

Table 1: The meaning of each field in `detReadout.dat`. The data in this file could be represented by a set of points in a graph with time on the x-axis and counts on the y-axis.

<*time*>	The time in hours at which the detector was read out
<*counts*>	The number of counts detected since the last readout
<*flag*>	This flag is 1 if the data is OK, 0 otherwise

The data is probably all background noise, but we want to be sure. Write a program that checks if, at any point, a significant number of subspace transmissions were detected. A point is considered significant if it is at least five standard deviations away from the mean. In case you forgot, the standard deviation of a data sample is given by

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^2\right) - \mu^2} \tag{1}$$

where N is the number of data points, x_i is the number of counts in the i:th point and μ is the average number of counts in the sample. Remember to check the data quality flag. If a point is not OK, it should not be included in the calculation. What values did you obtain for μ and σ?

Answer: The mean is $\mu = 24.96$, the standard deviation is $\sigma = 10.48$ and there is alien activity at 149h (130 counts), 486.4h (219 counts) and 653.8h (84 counts).

8 Write a class

The file `graph.cpp` contains a code skeleton for a rudimentary graphing class. Your task is to implement all of the missing functions. The file `test.cpp` contains a small program that creates a few graphs and prints the values returned by the various functions. Table **??** shows an example of the output you might get after completing the class. Numbers that are very close to but not exactly at their expected values are the result of small numerical errors.

Table 2: The table shows an example output for the four graphs used in `test.cpp`. Note that the x-axis used by the graph f(x) = $2x$ is logarithmic, such that the spacing between the points is much smaller at low x. When a value is outside the range defined by the graph, both the cases of returning zero and using a linear extrapolation are considered.

Graph Defined in	Empty -	(0, 1) -	f(x) = sin(x) [0, 2π]	f(x) = $2x$ [0.001, 10]
size	0	1	1000	1000
mean	0	1	8.91964e-14	2.1791
stdev	0	0	0.707107	4.14067
integral	0	0	1.99363e-05	99.999
Zero outside range				
f(-0.1)	0	1	0	0
f(pi/2)	0	1	1	3.14159
f(15)	0	1	0	0
Linear extrapolation outside range				
f(-0.1)	0	1	-0.0999993	-0.2
f(pi/2)	0	1	1	3.14159
f(15)	0	1	8.71641	30

You will have to make some design decisions while writing the class. For example, it is not immediately obvious what to do when the user wants to evaluate y at a point outside of the range defined by the graph. You could return zero, use the closest value or extrapolate somehow. Likewise, you should always make sure that the class knows what to do when graph is empty, contains only one point or other border cases. Such decisions come up frequently in real life, and you must use your best judgment. Compare your results to Table **??** and decide if they make sense.

9 Makefiles and inheritance

In this exercise, your task is to extend the `shape` class used in the lecture. Two additional classes should be implemented. The `circle` class should derive directly from `shape`. Its constructor should take the radius of the circle as an argument. The `square` class should derive from `rectangle`, and its constructor should take the side of the square as an argument. Implement getters and setters for the radius of the `circle` and the side of the `square`. You should then implement a new function `circumference` for all the various shapes. It should, as you guessed, return the circumference of the shape. Figure **??** shows the appearence of the `circle` and `triangle`. Notably, you may assume that the triangle is right-angled when calculating its circumference.

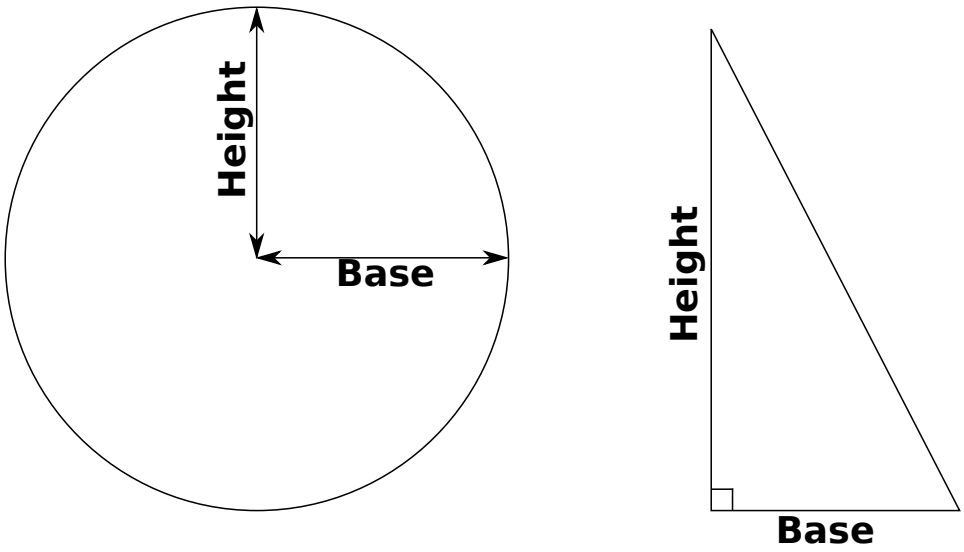

Figure 1: The appearence of the `circle` and `triangle` classes that derive from `shape`. The `base` and `height` of the `shape` class correspond to the radius of the circle. The `triangle` is assumed to be right angled.

The file `defaultTest.cpp` contains the example shown in the lecture. You can use it as a sanity check. The file `newTest.cpp` contains a program that will test the new shapes. Once you have implemented `circle` and `square`, uncomment the corresponding lines in `newTest.cpp` and confirm that the output matches what is shown in Table **??**.

Compiling `defaultTest.cpp` and `newTest.cpp` via command line is cum-

Table 3: The properties of the shapes used in `newTest.cpp`.

	Triangle	Rectangle	Square	Circle
Base	10	10	7	5
Height	5	5	7	5
Area	25	50	49	78.5
Circumference	26.2	30	28	31.4
Big enough	X	X	X	✓

bersome due to the many files involved. In the real world, a `make` tool is used for anything but the most trivial of programs. The `make` tool builds programs using information from a `Makefile`. Read through the provided `Makefile` and try to understand what it does. You can type `make defaultTest` or `make all` in a terminal to build the `defaultTest` executable. When you have implemented the new shapes, edit the `Makefile` so that it builds also the `newTest` executable.