**Area of a Triangle**

## *Heron’s Formula*

You will compute the area of a triangle using the famous Heron’s Formula. If we are given three sides of a triangle labeled as ***a***, ***b***, and ***c***, the area of that triangle can be computed as follows:

# *Area*=√(*s*(*s*−*a*)(*s*−*b*)(*s*−*c*))

*Figure 1: Heron's Formula*

In the above formula, the variable ***s*** represents the *Semiperimeter* of the triangle, computed as follows:

*s*=*a*+*b*+*c*

2

*Figure 2: Semiperimeter*

## Program Modules

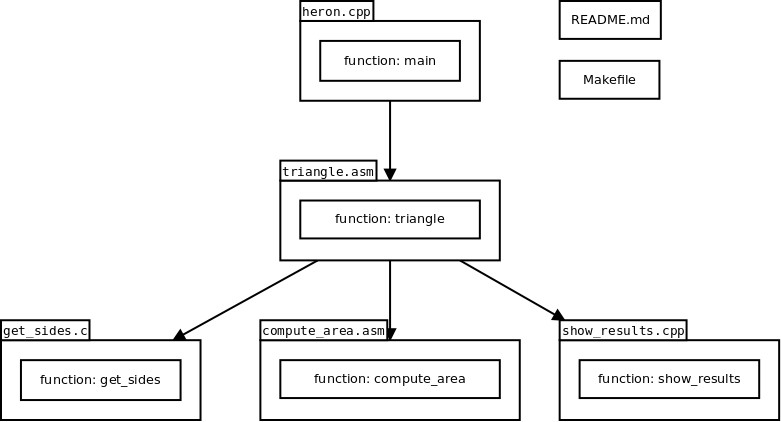
You will create several source files for this project:

|  |  |
| --- | --- |
| **File** | **Description** |
| README.md | This is a Markdown file containing your student information. See the *Code Submission Guidelines* document for more details. |
| Makefile | This is a GNU Makefile that will help you compile, link, and execute your program. It should compile your program piece by piece along the toolchain using multiple targets, as shown in class.  Each source file should compile to a corresponding object file. Each object file should have its own target which depends on that source file. An additional target that depends on all the object files should do the linking. Yet another target should run the program.  See the diagram below for a visual representation of Makefile target dependencies. |
| heron.cpp | This will be the entry point of your program (driver), which contains the main function. It is written in C++. It will print a welcome message, call on the assembly function inside triangle.asm, and print the result returned. |
| **File** | **Description** |
| triangle.asm | This module contains an assembly function called triangle, which manages finding the solution. It displays a message describing what this program will do. It will call the remaining three functions listed below, and eventually return a result back to the main driver function. |
| get\_sides.c | This module is fairly easy. It contains a C function who’s job is to ask the human to input the lengths of three sides of a triangle. It then returns those side lengths to the caller as three doubles (IEEE 754 64-bit floats).  The real problem is: How do you pass three values back to the caller? If you were solving this problem using 100% C, how would you return 3 numbers to the caller? What you would do in that case, is what you would do here.  (Hint: *Pass by reference*)  Also, this function should return true if all inputs were valid floats, or false if any inputs were invalid. This module shouldn’t print anything. |
| compute\_area.asm | This module contains the assembly function compute\_area, which receives 3 doubles representing the sides of a triangle. It uses Heron’s Formula to compute the area of that triangle, and returns it as a double to the caller. |
| show\_results.cpp | This module contains the C++ function show\_results, which will output the information we were seeking. |

To clarify, you will be using 64-bit IEEE 754 floating point numbers for all input, computations, and output.

### Module Diagram

The following diagram might help you understand the arrangement of the source files a bit more:



## Executable Name

Unless otherwise specified, the executable you generate should be named main for this and all future assignments.

## Makefile Behavior

When invoking the default target (i.e., the user types “make” with no arguments), your Makefile should compile and link your program and produce the executable.

You should leverage the power of Makefile dependencies such that multiple calls to the default target will recompile only the modules that need to be recompiled (based on source code modification), and relink your program only if it does not exist or one of the modules has changed.

The default target should ***not run*** your program, but merely compile + link (as applicable).

## Example Runs

The easiest way to describe this program is probably with sample output.

You should study the following examples until your own program can produce identical output. Each module in your program has a different job, so may produce different output (or request input). The examples below will have each module’s output labeled as such:

1. Heron (driver)
2. triangle
3. get\_sides
4. show\_results

Note that the output shown for get\_sides (3) is actually the user’s input, not anything printed by get\_sides.

### Example 1

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  4  2  1 |
| Welcome to CIS/CSC-11, Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 1  2  2 |
|  |
| The area of a triangle with sides 1.0000000000, 2.0000000000, and 2.0000000000 is 0.9682458366 square units. |
|  |
| The area will be returned to Heron. |
|  |
| Heron received this number: 0.9682458366.  Have a nice day. The program will return control to the operating system. |

### Example 2

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  4  2  1 |
| Welcome to CIS/CSC-11, Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 7.3378  12.28744  6.37637 |
|  |
| The area of a triangle with sides 7.3378000000, 12.2874400000, and 6.3763700000 is 18.6525801599 square units. |
|  |
| The area will be returned to Heron. |
|  |
| Heron received this number: 18.6525801599.  Have a nice day. The program will return control to the operating system. |

### Example 3

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  4 |
| Welcome to CIS/CSC-11 Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 18237687236.8762359619  12786837464.8273677826  13987298733.8768768311 |
|  |
| The area of a triangle with sides 18237687236.8762359619, |
| ***Output*** | ***Module***  2  1 |
| 12786837464.8273677826, and 13987298733.8768768311 is 89180278794645405696.0000000000 square units. |
|  |
| The area will be returned to Heron. |
|  |
| Heron received this number: 89180278794645405696.0000000000.  Have a nice day. The program will return control to the operating system. |

### Example 4

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  2  4  2  1 |
| Welcome to CIS/CSC-11 Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 123.2760000000  10.3287000000  2.2765000000 |
|  |
| Your triangle is nonsense! |
|  |
| The area of a triangle with sides 123.2760000000, 10.3287000000, and 2.2765000000 is 0.0000000000 square units. |
|  |
| The area will be returned to Heron. |
|  |
| Heron received this number: 0.0000000000.  Have a nice day. The program will return control to the operating system. |

### Example 5

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  2  2  1 |
| Welcome to CIS/CSC-11 Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 123.2760000000  GARY |
|  |
| Your input is nonsense! |
|  |
| The area will be returned to Heron. |
|  |
| Heron received this number: 0.0000000000.  Have a nice day. The program will return control to the operating system. |

### Example 6

(note that during module 3, the user enters a blank line for input)

|  |  |
| --- | --- |
| ***Output*** | ***Module***  1  2  3  2 |
| Welcome to CIS/CSC-11 Area of a Triangle brought to you by Wisteria Ravenclaw. |
|  |
| The manager is here to help you find the area of your triangle.  Input your 3 floating point numbers representing the sides of a triangle. Press enter after each number. |
|  |
| 123.2760000000 |
|  |
| Your input is nonsense! |
| ***Output*** | ***Module***  2  1 |
| The area will be returned to Heron. |
|  |
| Heron received this number: 0.0000000000.  Have a nice day. The program will return control to the operating system. |

## Special Issues

This section contains instructions for special issues you should account for.

### Nonsense Triangles

Sometimes the three inputted numbers are errors. That is, there is no such triangle that can be made from the entered side lengths.



For example, there is no such triangle with side lengths of 1, 1, and 10. Imagine what that would look like, for a moment. You can detect this somewhere during your calculations because Heron’s formula will attempt to take the square root of a negative number.

In such cases of “no such triangle”, do the following:

* Your computed area should be zero.
* Before printing the computed area line, print a message about the triangle being nonsense. Do not print this message from any module but the *triangle* module. This is the message:

◦ Your triangle is nonsense!

### Floating Point Precision

When you are using C++ to print floating point numbers, make sure to use fixed precision with 10 numbers printed after the decimal point. The C++ objects *std::fixed* and *std::setprecision* can help you achieve this, and are found inside the *iomanip* include header.

For example, a properly formatted value of 10 would actually look like the following:

• 10.0000000000

### Stack Alignment

You may notice mysterious segmentation faults in your C module, when printing floats with libP, or elsewhere. This is sometimes caused by stack alignment issues. Essentially, your stack is expected to be aligned to 16 bytes when calling functions in other libraries. You may wish to experiment pushing an extra register to the stack before making such calls, just to see if that fixes your issues.

### Passing by Reference in C

Regular C modules can’t pass function arguments by reference. If your module works with reference arguments, you may have accidentally compiled a C++ program.

You can accomplish the same thing in C code by passing a *pointer* to a variable, instead of the actual value. Here’s an example you may wish to use in your *get\_sides* module:

void get\_sides(double\* side1, double\* side2, double\* side3)

### Inputting Floats in C

You also cannot use *cin* in a C program. If your module is able to use *cin* or any other classes/objects, you may have accidentally compiled a C++ program.

In C, you can input a double precision floating point value from the user with the following snippet of code:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h> #include <limits.h> int main () {  char input[LINE\_MAX]; double d; int returnValue;  /\*  Simulate Input This section skips the step of grabbing input from the user.  We simulate user input by copying a string directly into the input  buffer, in order to quickly show how sscanf would behave.  \*/  // Simulate a valid input strcpy( input, "71.82348" ); |

|  |
| --- |
| returnValue = sscanf( input, "%lf", &d ); printf("%d ==> %lf\n", returnValue, d);  // Simulate another valid input  strcpy( input, "-73278923651.8235587652348" ); returnValue = sscanf( input, "%lf", &d ); printf("%d ==> %lf\n", returnValue, d);  // Simulate an invalid input strcpy( input, "q123.376" );  returnValue = sscanf( input, "%lf", &d ); printf("%d ==> %lf\n", returnValue, d);  // Simulate another invalid input strcpy( input, "1287634.2876q" ); returnValue = sscanf( input, "%lf", &d ); printf("%d ==> %lf\n", returnValue, d);  // Simulate another invalid input  strcpy( input, "q" );  returnValue = sscanf( input, "%lf", &d ); printf("%d ==> %lf\n", returnValue, d);  /\* Actually get input from the user.  This shows a complete example of how you could take input in a  program.  Note: This is just an example and is not meant to be copy/pasted or  taken literally.  For example, printing "Please enter a float" is probably not  part of any assignment.  \*/  printf("Please enter a float: ");  if (fgets(input, LINE\_MAX, stdin) == NULL) { printf("You entered no line!\n");  } else {  // Will return 0 for invalid, -1 for blank input returnValue = sscanf( input, "%lf", &d );  if ( returnValue < 0 ) { printf("Your input was blank!\n");  }  else if ( returnValue == 0 ) { printf("You entered nonsense!\n");  } else{ printf("You entered a valid float: %lf (returnValue = %d)\n",  d, returnValue);  }  } |

return(0); }

Note that if your double is already a pointer, using the ampersand (&) in sscanf above would not be needed (hint).

Don’t forget to include the appropriate headers.