CSE 2421 – Systems 1

Spring Semester 2024

Programming Assignment #1

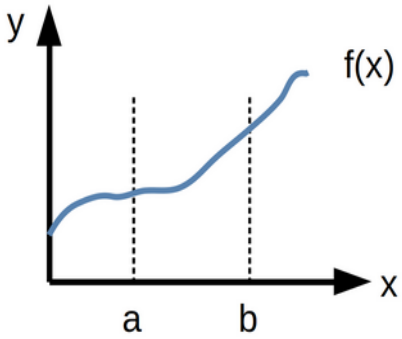
- **This assignment is worth 13pts.**
- **You must upload the zipped solution folder to Carmen, as solution.zip.**
- **The deadline for this assignment is January 31st 11:59pm ET.**
- **Deductions for late submissions apply.**

This assignment consists on approximating the area of the square function, f(x) = x*x, by using Riemann Sums (https://en.wikipedia.org/wiki/Riemann_sum).

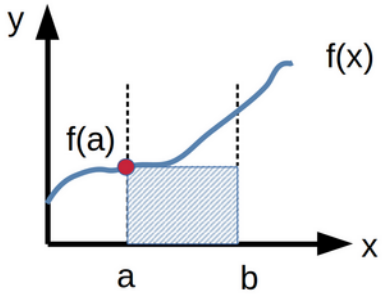
The key idea is to approximate the area under the f(x) curve, within a specific interval of x, by computing the sum of the area of a finite number of rectangles.

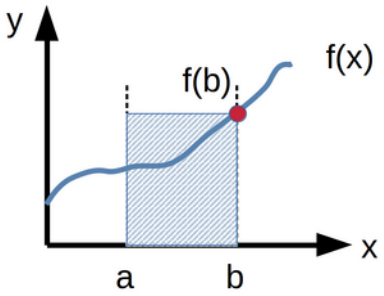
For reference, recall that the area under the function x*x, with x in [a,b] can be computed with: $(b^3 – a^3) / 3.0$.

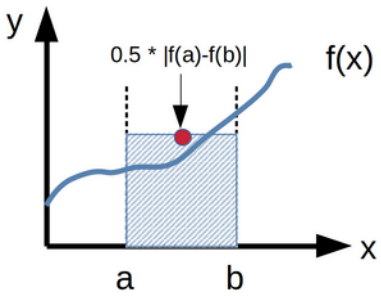
To illustrate how the algorithm you will implement works, consider the following figure:



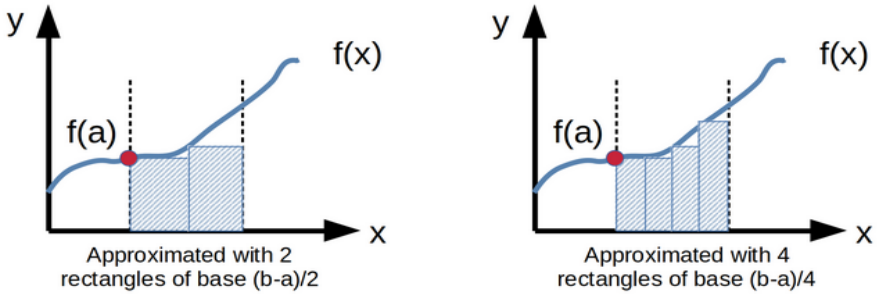
The area under the curve can first be approximated with the area of a single rectangle of base (b-a) and height h, and where h can be either of three possible heights: f(a), f(b) or |f(b)-f(a)|/2:







This first approximation is far from ideal. However, it can be improved by increasing the number of rectangles used to approximate the area under the curve f(x). Consider the following two scenarios, where 2 and 4 rectangles are used instead of a single one:



If the number of rectangles being added is large enough, the area under function f(x) will be sufficiently approximated by the sum of the areas of the rectangles.

Initial Instructions:

- Download the entire folder a1 from Carmen.
- You can debug and do the preliminary implementation of your code in your own laptop or desktop, but the final testing of your code must be done and work in stdlinux or coelinux.
- After downloading the folder a1 (or the file a1.tar.gz), copy the file/folder to your favorite OSU linux cluster (stdlinux or coelinux).
- Write your code exclusively in the file area-x2.c
- **Do not modify the main function in area-x2.c**. Doing so could produce incorrect results for your code.
- After transferring the entire directory (or tar.gz file) to the remote host, add execution permissions to the script run-tests.sh by doing in the command line: chmod +x run-tests.sh

Program Criteria and Grading.

You are asked to write a program that meets the following criteria:

- The function area_x2 should compute the approximate area under x^2 that meets the required precision. area_x2 will take three arguments and match the invocation from the main function.
- The criterion to determine that the area under the curve has been approximated by a large enough number of rectangles is: | area(a,b,n_intervals) – area(a,b,n_intervals+1) | < precision
- **[1 pt]** Your program must use at least 3 user-defined functions, in addition to the main routine.
- You should use only double and int data types.
- As height for each rectangle you must use h = | f(left) – f(right) | / 2.0, where left and right are the bounds of a single rectangle being used.
- All rectangles must have the same width as base.
- Your program should compile without warnings. You must use the compiler flags:
  -pedantic -O0 -Wformat -Wreturn-type -lm
- **[1 pt]** All your variables and functions should have meaningful names.
- **[1 pt]** Your code should be documented. Each function should have at least two useful and descriptive comments.
- Invalid inputs: The function area_x2 invoked from main function must handle two special cases of inputs. When the given precision is <= 0 and whenever the interval [a,b] is not well-defined, that is, when b <= a. In such cases your program should print negative values for all variables. It's recommended to use a **goto with a label** to handle these corner cases.
- **[1 pt]** Declaring the headers of all functions: All function headers must appear between the last include directive and the first defined function.
- **[8 pt total : 1 pt each]** Your output should match that of the file reference.txt
- **[1 pt]** Include the required screen capture to show the date/time and your username with the produced output. See instructions below.

A few warnings and recommendations:

- It may be the case that when testing your code your program will hang. If that happens, use 'Ctrl+C' to cancel the execution of the binary from the command line.
- Use run-tests.sh only when you believe your program is fully correct and ready to test for all cases. Before that you should be testing with your own individual inputs and possibly using the python script provided.
- To test your program when debugging, work in the command line and test individual cases (one line of inputs.txt).
- The graders may use a different and complementary set of inputs and outputs. You are free to use additional test cases for your own debugging.
- It is recommended to not manually delete any files to avoid losing and or all your work. The scripts provided will do almost all the work for you.

For your convenience, you are provided with the following files:

- area-x2.c: The c file in which you must write your code.
- run-tests.sh: A bash script to compile your program, build it, run it, and test several inputs. You will observe if you
- area-square.py: A python script to compute the area under the function f(x)=x*x within the interval [a,b]. The script will compute and return $x^3/3.0$ (the integral of x^2). To use the python script, type in your command line:

  python ./area-square.py a b

  For example, the script will print 2.3333333333333335 when invoked in stdlinux, from the command line, via:

  python area-square.py 1.0 2.0

  If testing in coelinux, replace "python" by "python3" in the above command.
  The provided python script can be used to debug individual valid inputs and determine, more or less, what you should be producing as approximated area.
- Makefile: You can build your program and the test harness with it, as seen in class.
- inputs.txt: A text file with several inputs to be used for testing and evaluation of your code.
- reference.txt: A text file with reference output. It was generated by a compliant implementation of the program. Do not modify this file.
- check-output.c: A test harness which will be used to validate the output of your program. DO NOT CHANGE ANYTHING IN THIS FILE. The graders will use a freshly downloaded copy of it when validating your solution.

What to upload:

- Copy your working file area-x2.c, where you wrote your code, to the folder "solution". You must do this right before uploading your code.
- Test your code in stdlinux or coelinux with run-tests.sh. Immediately after testing your working code, take a screenshot of your command line. It should show you logged into some remote node of either stdlinux or coelinux as well as the timestamp and username. The screenshot should be taken when you are sure your code is complete, correct, and ready to submit. Copy the screenshot file (which will be in your local machine, not in the remote machine) to the solution folder (which is probably still on the remote host).
- Zip the solution folder and upload the zipped folder, solution.zip, to Carmen.

Where to upload:

Upload via Assignments->Assignment 1 in Carmen.
Do not confuse with Files->Assignments.