Branching and Repetition Lab

Note: This work must be completed within 1 week of this lab time.

Name:	ID:

Goals:

1. Be able to use branching and repetition statements in C.

Part A:

- The freezing point of water decreases as the salinity increases. So if you have two containers of liquid, one fresh, the other salt, as the temperature decreases, the fresh water will freeze before the salt water does.
- Given a number of data points (freezing points and salinity pairs), it is possible to use linear interpolation to determine what the freezing point of a particular water sample will be if the salinity changes:

```
f(b) = f(a) + [f(c) - f(a)] * (b - a) / (c - a)
where:
    f(b) = the estimated value for b
    a, f(a) = first known data point
    c, f(c) = second known data point
and:
    a < b < c</pre>
```

• To make the use of the formula a bit clearer, the example below uses temperatures in °F. Given that fresh water freezes at 32°F at sea level, and salt water with a salinity of 10ppt freezes at 30.1°F, salt water with a salinity of 5 ppt freezes at:

```
f(b) = 32 + [30.1 - 32] * (5 - 0) / (10 - 0)
= 32 - 1.9 * 5 / 10 = 31.05
```

- Write a C program that prompts for a, f(a), c, f(c), and b and prints out the corresponding f(b)
- Exit gracefully (using an error message and a return -1) unless a < b < c

Part B:

- Write a C program that prompts the user for a number of data points x_i and $f(x_i)$ and then prints out the corresponding bar graph.
- A sample output is below:

```
Enter the number of data points: 2
Enter x[0]:
1
Enter f(x[0]):
```

```
5
1: ****
Enter x[1]:
2
Enter f(x[1]):
10
2: *******
```

- If the user enters an invalid number of data points, keep prompting until a value >= 1 has been entered.
- Data points x_i and $f(x_i)$ can be negative. In case $f(x_i)$ is negative, print N/A instead of a bar graph. e.g., if $x_i = -1$ and $f(x_i) = -10$, print -1: N/A

Deliverables:

- Submit printouts of both programs (Part A and Part B), stapled to this lab sheet
- After submitting the printouts, call your instructor to demonstrate your programs. You will be given several data points to try out, in order to test the correctness of your programs.
- You will not be able to make any changes once you have submitted the printouts, so be sure to test your programs thoroughly before submitting them!

Instructor Use for Part A:	
Test Result 1:	or □Graceful Exit or □Other Error or □Crash
Test Result 2:	or □Graceful Exit or □Other Error or □Crash
Test Result 3:	or □Graceful Exit or □Other Error or □Crash
Test Result 4:	or □Graceful Exit or □Other Error or □Crash
Instructor Use for Part B:	
Test Result 1:	or \square Graceful Recovery or \square Other Error or \square Crash
Test Result 2:	or \square Graceful Recovery or \square Other Error or \square Crash
Test Result 3:	or \square Graceful Recovery or \square Other Error or \square Crash
Test Result 4:	or □Graceful Recovery or □Other Error or □Crash

- passes all tests - passes all tests - well-documented, allowing another programmer to use all functions based on the header comments alone - code review reveals no faults - code review reveals no faults - efficient (given the requirements) - all parameters, return values, and side effects are explained - no redundant operations - comments within all functions are helpful, without being distracting, making it easy to follow along - no indestrables appropriately - no layout abnormalities (egimproper indentation) - casily used and reused - no undesirable side effects output - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor cases - at most a few redundancies - a code review reveals nearly no faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - of no an number of major probe which render the work unuss.	er appropriate s unavoidable are named g, missing or
well, without giving implementation details - efficient (given the requirements) - all parameters, return values, and side effects are explained - no redundant operations - comments within all functions are helpful, without being distracting, making it easy to follow along - no layout abnormalities (egimproper indentation) - easily used and reused - no undesirable side effects, output - passes nearly all tests - occasionally, there are comments that are not complete, helpful, and/or true - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor - at most a few redundancies - a code review reveals a number of faults, but none, or almost none of them, are major - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - somewhat efficient, but there are a small number of major inefficiencies - often easily used and reused issues, none, or almost none small number of major inefficiencies - often easily used and reuse small number of major prob	s unavoidable are named s, missing or
effects are explained - no redundant operations - comments within all functions are helpful, without being distracting, making it easy to follow along - no layout abnormalities (egimproper indentation) - assily used and reused - no undesirable side effects, output - passes nearly all tests - occasionally, there are comments that are not complete, helpful, and/or true - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor cases - at most a few redundancies - at most a few redundancies - at most a few redundancies - a toode review reveals a number of faults, incomplete, misleading, missing, or redundant - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - comments within all functions are helpful, and present appropriately - no layout abnormalities (egimproper indentation) - a largely well-engineered, exeminor insures or a few, minor issues with complete, helpful, and/or true - a few, minor issues with complete or a few redundancies - a timest a few redundancies - at most a few redundancies - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant with constants, variables, or one of major i	are named
without being distracting, making it easy to follow along - no layout abnormalities (eximproper indentation) - easily used and reused - no undesirable side effects, output - passes nearly all tests - occasionally, there are comments that are not complete, helpful, and/or true - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor cases - at most a few redundancies - passes half the tests, or more, but the failure rate is too high for a 4-star rating - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - often easily used and reused, except in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of occasions, with constants, variables, or or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - often easily used and reused. - on a number of occasions, with constants, variables, or or almost none of them, are major	g, missing or
improper indentation) - easily used and reused - no undesirable side effects, output - passes nearly all tests - occasionally, there are comments that are not complete, helpful, and/or true - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor cases - at most a few redundancies - passes half the tests, or more, but the failure rate is too high for a 4-star rating - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - occasionally, there are comments that are not complete, helpful, and/or true - a code review reveals nearly no faults; faults - could be improved by slightly reworded, slightly reworded, slightly more, or slightly fewer comments - a few, minor issues with con layout - a few, minor issues with con layout - a sily used and reused, excending in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of occasions, with constants, variables, or - on a number of occasions, with constants, variables, or	
*** good - passes nearly all tests - occasionally, there are comments that are not complete, helpful, and/or true - largely well-engineered, exminor issues - a code review reveals nearly no faults; faults that are found, are minor - cases - could be improved by slightly reworded, slightly more, or slightly fewer comments - a few, minor issues with coor layout - easily used and reused, exeminor instances - at most a few redundancies - generally no undesirable si *** ok - passes half the tests, or more, but the failure rate is too high for a 4-star rating - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of occasions, with constants, variables, or - on a number of occasions, with constants, variables, or - often easily used and reuse small number of major inefficiencies	
**** good	
complete, helpful, and/or true minor issues - a code review reveals nearly no faults; faults that are found, are minor - generally efficient, except in a few minor cases - at most a few redundancies - passes half the tests, or more, but the failure rate is too high for a 4-star rating - a code review reveals a number of faults, but none, or almost none of them, are major - a code review reveals a number of faults, but none, or almost none of them, are major - a code review reveals a number of faults, but number of major inefficiencies - a code review reveals a number of faults, but number of major inefficiencies - a code review reveals a number of faults, but number of major inefficiencies - a code review reveals a number of faults, but number of major inefficiencies - a code review reveals a number of faults, but number of major inefficiencies - a code review reveals a number of faults, but number of major inefficiencies	such as debug
that are found, are minor slightly more, or slightly fewer comments or layout - generally efficient, except in a few minor cases - at most a few redundancies - at most a few redundancies - passes half the tests, or more, but the failure rate is too high for a 4-star rating - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies slightly more, or slightly fewer comments - easily used and reused, exception in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - in need of reengineering due issues, none, or almost none major - on a number of occasions, with constants, variables, or - often easily used and reuse small number of major prob	cept for a few,
cases minor instances - at most a few redundancies - generally no undesirable si *** ok - passes half the tests, or more, but the failure rate is too high for a 4-star rating - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - a code review reveals a number of faults, but none, or almost none of them, are major - on a number of occasions, with constants, variables, or - often easily used and reuse small number of major inefficiencies - on a number of major probi	nstants, variables,
*** ok - passes half the tests, or more, but the failure rate is too high for a 4-star rating - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant - on a number of occasions, with constants, variables, or - often easily used and reuse small number of major inefficiencies	ept in a few,
rate is too high for a 4-star rating - a code review reveals a number of faults, but none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies - somewhat of major inefficiencies - passes main the ways, or maintenance of casts, comments at crypto, false, incomplete, misleading, missing, or estimated or easily assess, none, or almost none major - on a number of occasions, with constants, variables, or - often easily used and reuse small number of major prob	de effects
none, or almost none of them, are major - somewhat efficient, but there are a small number of major inefficiencies with constants, variables, or - often easily used and reuse small number of major probi	
number of major inefficiencies small number of major probi	
	ems, none of
- potentially many redundancies	
** - fails more than half the tests - only little relevant documentation - a number of major issues, the changes	equiring major
- a code review reveals a high number of faults, including a number of major faults are often cryptic, false, incomplete, misleading, missing, or redundant - not easily used and reused	
- not efficient, although slow progress is being made	
* - fails nearly all the tests - essentially no legitimate documentation - only a few functions, many responsible for too many thi	
- code review reveals a proliferation of major faults - essentially not usable or re-	
0 fail - fails all tests and/or does not run - no legitimate documentation and/or does not run - no legitimate code and/or or	ngs

* fail	- fails nearly all the tests	- essentially no legitimate documentation	- only a few functions, many of which are responsible for too many things
	- code review reveals a proliferation of major faults		- essentially not usable or reusable
0 fail	- fails all tests and/or does not run	- no legitimate documentation and/or does not run	- no legitimate code and/or does not run
Documer	,	x 2 = / 10	Subtotal A: / 20

Rating	Correctness/Efficiency	Documentation	Structure/Complexity
***** perfect	- passes all tests	- well-documented, allowing another programmer to use all functions based on the header comments alone	- well-engineered, consisting of a modular collection of simple, single-purpose functions
	- code review reveals no faults	- responsibilities of all functions are described well, without giving implementation details	- constants are used whenever appropriate
	- efficient (given the requirements)	- all parameters, return values, and side effects are explained	- globals are not used, unless unavoidable
	- no redundant operations	- comments within all functions are helpful, without being distracting, making it easy to follow along	- all constants and variables are named appropriately
			- no layout abnormalities (eg, missing or improper indentation)
			- easily used and reused
			- no undesirable side effects, such as debug output
**** good	- passes nearly all tests	- occasionally, there are comments that are not complete, helpful, and/or true	- largely well-engineered, except for a few, minor issues
	- a code review reveals nearly no faults; faults that are found, are minor	- could be improved by slightly reworded, slightly more, or slightly fewer comments	- a few, minor issues with constants, variables, or layout
	- generally efficient, except in a few minor cases		- easily used and reused, except in a few, minor instances
	- at most a few redundancies		- generally no undesirable side effects
*** ok	- passes half the tests, or more, but the failure rate is too high for a 4-star rating	- in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant	- in need of reengineering due to a number of issues, none, or almost none of which, are major
	- a code review reveals a number of faults, but none, or almost none of them, are major		- on a number of occasions, there are issues with constants, variables, or layout
	- somewhat efficient, but there are a small number of major inefficiencies		- often easily used and reused, but there are a small number of major problems, none of which render the work unusable
	- potentially many redundancies		
** fail	- fails more than half the tests	- only little relevant documentation	- a number of major issues, requiring major changes
	- a code review reveals a high number of faults, including a number of major faults	- comments are often cryptic, false, incomplete, misleading, missing, or redundant	- not easily used and reused
	- not efficient, although slow progress is being made		
* fail	- fails nearly all the tests	- essentially no legitimate documentation	- only a few functions, many of which are responsible for too many things
	- code review reveals a proliferation of major faults		- essentially not usable or reusable
0 fail	- fails all tests and/or does not run	- no legitimate documentation and/or does not run	- no legitimate code and/or does not run

Correctness/Efficiency:	/ 5 x Z = / 10		
Documentation:	/ 5	Subtotal B:	/ 20
Structure/Complexity:	/ 5	Total: / 40 =	/ 10