Midterm Exam

For all questions, validate data where appropriate.

1) In folder 01_expressions write the function prototype in .h file and function definitions in .cpp for clock 3 functions get_hours, get_minutes, and get_seconds. Write the required unit test(s) in folder 01_expressions_test.

   Difficulty Level - Easy 5 points

   NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

   Implementation: 1.75 points
   a) Write a function get_hours that returns an int and accepts an int seconds_since_1970 parameter
   b) Write a function get_minutes that returns an int and accepts an int seconds_since_1970 parameter
   c) Write a function get_seconds that returns an int and accepts an int seconds_since_1970 parameter

   Test Cases: 1.75 points
   Test 1
   Function argument: 1570846218
   function get_hours should return 2
   function get_minutes should return 10
   function get_seconds should return 18

   Test 2
   Function argument: 1570875018
   function get_hours should return 10
   function get_minutes should return 10
   function get_seconds should return 18

   Main program flow: 1.5 points
   No loop. Use provided seconds since 1970 value as argument to get_hours, get_minutes, and get_seconds function accordingly and display the time.

2) In folder 02_decisions_if, in .h file write the function prototype which accepts two strings. In .cpp file code definition for the p-distance, proportion of different characters of two dna strings. Write the required unit test(s) in folder 02_decisions_if_test.

   Difficulty Level – Easy 5 points

   NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

   Implementation: 1.75 points
   Write function get_dna_p_distance with two const reference string function parameters which returns the ratio(double) of characters that differ in parameter1 and parameter2.
   Example:

double p_distance = get_dna_p_distance("GAGCCTACTAACGGGAT", "CATCGTAATGACGGCCT");
distance will be 7/17 = .4118
Round your answers to four decimals spaces.

Test Cases: 1.75 points
for the function get_dna_p_distance test with string parameters:
a) "GAGCCTACTAACGGGAT", "CATCGTAATGACGGCCT" the results should be .4118.
b) "GAGCCTACTAACGGGAT", "GATCGTAATGACGGCCT" the results should be .3529.

Main program flow: 1.5 points
No loop.
Use these strings "GAGCCTACTAACGGGAT", "CATCGTAATGACGGCCT" as function arguments and display the p-distance of the strings.

3) In folder 03_decisions_switch, in .h file write prototype for string value - return function gpa_to_letter_grade with a double parameter that returns the letter grade(string) of that GPA. In .cpp write function implementation code. Write the required unit test(s) in folder 03_decisions_switch_test.

Difficulty Level – Easy 5 points

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Implementation: 1.75 points
For function gpa_to_letter_grade write code to return a letter grade given a GPA.

Given a double 3.5 returns the string A
TIP: You'll have to convert the double to an int using multiplication

GPA to letter grade conversion
3.50 to 4.00 returns A
3.00 to 3.49 returns B
1.70 to 2.99 returns C
1.00 to 1.69 returns D
less than 1 returns F

Test Cases: 1.75 points
For function gpa_to_letter_grade write test case as follows:
3.50 returns an "A"
3.25 returns a "B"
2.99 returns a "C"
1.69 returns a "D"
.5 returns an "F"

Main program flow: 1.5 points
No loop. Prompt the user for one gpa and use it as the function argument to call the gpa_to_letter_grade function and display the letter grade to screen.

4) In folder 04_loops_simple_data inf .h file write prototype for function get_fibonacci with an int parameter that returns the fibonacci sequence(string) up to that number. In .cpp write function implementation code. Write the required unit test(s) in folder 04_loops_simple_data_test.

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Difficulty Level – Intermediate 12.5 points

Implementation: 4.5 points
Function get_fibonacci returns the Fibonacci sequence string of a number.
Example: get_fibonacci(5) returns "0, 1, 1, 2, 3, 5"
Do not copy the double quotes to Visual Studio, your program will generate errors.

Test Cases: 4.5 points
Write test case for get_fibonacci with function argument 10:
result "0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55"
Write test case for get_fibonacci with function argument 10:
result "0, 1, 1, 2, 3, 5"

Main program flow: 3.5 points
Program runs until user opts out.
For each loop prompt user for a number, use number as function argument, call get_fibonacci function and display the output.

5) In folder 05_loops_strings write prototype and definition for string value - return function transcribe_dna_into_rna with a string parameter that returns the rna string. Write the required unit test(s) in folder 05_loops_strings_test

.
Difficulty Level – Intermediate 12.5 points

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Implementation: 4.5 points
An RNA string is a string formed from the alphabet containing 'A', 'C', 'G', and 'U'.
Given a DNA string t corresponding to a coding strand, its transcribed RNA string u is formed by replacing all occurrences of 'T' in t with 'U' in u.
Example:

transcribe_dna_into_rna("GATGGAACTTGACTACGTAAATT");
returns:
GAUGGAACUUGACUACGUAAAUU

Test Cases: 4.5 points
Write test case for transcribe_dna_to_rna with string:

"GATGGAACTTGACTACGTAAATT" returns “GAUGGAACUUGACUACGUAAAUU”
Do not copy the double quotes to Visual Studio, your program will generate errors.

Main program flow: 3.5 points
No loop. Use the given string as function argument call for transcribe_dna_into_rna function and display the output.

6) In folder 06_loops_vectors in .h file write prototype function get_dna_p_distance_vector with a const reference vector of string parameter which returns vector of double value. In .cpp write the function implementation code. Write the required unit test(s) in folder 06_loops_vectors_test.

Difficulty Level – Difficult 20 points

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Implementation: 10 points

For a general distance function dna strings s1,s2,…,sn , we may encode the distances(p-distance) between pairs of dna strings via a distance matrix D in which Di,j=d(si,sj).

For two strings s1 and s2 of equal length, the p-distance between them is the proportion of corresponding symbols that differ between s1 and s2.
==THIS has been done in question 2, use the function get_dna_p_distance from question 2 in the function get_dna_p_distance_vector=== (Make sure to include the dna_p_distance.h(Question 2) header file to the dna_p_distance_vector.cpp file.)

FUNCTION get_dna_p_distance_vector

Given: A collection, vector of string, of n (n≤10) DNA strings s1,…,sn of equal length.

Function argument Sample vector of string

```
 vector<string> data {"TTTCCATTTA",
                      "GATTCATTTC",
                      "TTTCCATTTT",
                      "GTTCCATTTA"};
```

```
vector<double> result;
```

Algorithm:

```
for each string in data vector
        get p distance for current string compared to itself and all the other 3 strings
        save its rounded p-distance return value from get_dna_p_distance function to the result vector
```

Return: The vector of doubles corresponding to the p-distance dp on the given strings.

Sample return vector of doubles

```
{
0.00000, 0.40000, 0.10000, 0.10000,
```

0.40000, 0.00000, 0.40000, 0.30000,
0.10000, 0.40000, 0.00000, 0.20000,
0.10000, 0.30000, 0.20000, 0.00000
}

Explanation for first row of return vector:
compare "TTTCCATTTA" to itself to return get_dna_p_distance value of 0 (all characters are the same)
compare "TTTCCATTTA" to "**GA**T**T**CATTT**C**" get_dna_p_distance value of .4 (4 characters differ)
compare "TTTCCATTTA" to "TTTCCATTT**T**" get_dna_p_distance value of .1 (1 character differs)
compare "TTTCCATTTA" to "**G**TTCCATTTA" get_dna_p_distance value of .1 (1 character differs)

Test Cases: 3 points
Write test case for get p distance vector with vector string value
{"TTTCCATTTA", "GATTCATTTC", "TTTCCATTTT", "GTTCCATTTA"};
returns a vector of double
{
0.00000, 0.40000, 0.10000, 0.10000,
0.40000, 0.00000, 0.40000, 0.30000,
0.10000, 0.40000, 0.00000, 0.20000,
0.10000, 0.30000, 0.20000, 0.00000
}

Main program flow: 2 points
No loop. Use the given vector of string data as function argument, call get_dna_p_distance_vector function and display the output.

Memory Diagram: 5 points
Upload to folder 06_loops_vectors Github
Draw the memory diagram for your main.cpp program statements. To represent the strings in memory use 10 byte increments in the addresses. **Don't draw a diagram for get_dna_p_distance_vector function internals.**
Example:
first address: 00 TTCCATTTA
next address: 10 other string

7) In folder 07_function_value_params in .h file write prototype void function value_params with a value vector of int parameter. In .cpp file write function code for ranged loop with auto value variable, change the value in the for loop code to some other number. Write the required unit test(s) in folder 07_function_value_params_test.

Difficulty Level – Intermediate 10 points

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Implementation: 3 points
For function value_params write a for ranged loop with auto value variable, change the value in the for loop code to some other number

Test Cases: 3 points
Write test case for get_value_params with vector of int {2,3, 5, 7} function argument.
Show that the vector of int changed or didn't change

Main program flow: 1
No loop. Use the vector of int values as function argument for value_params function, call function and display the vector of int to show that the vector changed or didn't change.

Memory Diagram: 3 points
Upload to folder 07_function_value_params in GitHub
Draw the main.cpp memory diagram to show what happens when vector of int variable is created and used in the value_params function as an argument. Also include what happens in the function for ranged loop.

8) In folder 08_function_ref_params write prototype and definition for void function ref_params with a reference vector of int parameter. In the function use a for ranged loop with auto reference variable, change the variable value in the for loop code to some other number. Write the required unit test(s) in folder 08_function_reg_params_test.

   Difficulty Level – Intermediate 10 points

   NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

   Implementation: 3 points
   use a for ranged loop with auto reference variable, change the variable value in the for loop code to some other number

   Test Cases: 3 points
   Write test case for get_value_params with vector of int {2,3, 5, 7} function argument.
   Show that the vector of int changed or didn't change

   Main program flow: 1 point
   No loop. Use the vector of int values as function argument for ref_params function, call function and display the vector of int to show that the vector changed or didn't change.

   Memory Diagram: 3 points
   Upload to folder 08_function_ref_params in Github
   Draw the main.cpp memory diagram to show what happens when vector of int variable is created and it's used in the ref_params function as an argument.

9) In folder 09_class_basics in .h file write the Dive class interface. In .cpp write the implementation code for the class functions. Write the required unit test(s) in folder 09_class_basics_test.

   Difficulty Level – Difficult 20 points

   NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

   Implementation: **10 points**
   Write a class Dive which contains class data as follows: start pressure, finish pressure, time and depth. Typical

values are a starting(**s**) pressure of 3000, ending(**e**) pressure of 700, depth(**d**) of 30 to 80 feet and times(**t**) of 30 minutes (at 80 feet) to 60 minutes (at 30 feet). SACR's are typically between 10 and 20.
Formula for SACR calculation : sacr = 33(s - f) / t(d + 33)

DO NO USE PUBLIC CLASS MEMBER VARIABLES
What constructor is required?

Determine which variables need to be created

Class should have a class function named get_sacr() which returns the SACR for a dive.

Test Cases: 5 points
Test case for Dive class
Create an instance of Dive class.
The Dive class data member are:
d = depth in feet = 55
s = starting pressure = 3000
f = final pressure = 1000
t = time in minutes = 30
result should return 25

Main program flow: 5 points
Loop continues until user opts out. Prompt the user for starting pressure, ending pressure, depth (make sure the allowable values are between 30 and 80 only), and time in minutes. Create a Dive class instance with user-data, call the get_sacr function and display the value to screen:
Sample output:
SACR: 10.59

10) In folder 10_class_copy convert question 1 clock functions to a class Clock implementation. Create the proper private variables and public function to display_time. THERE IS NO NEED TO USE OVERLOADED OPERATORS.

__Difficulty Level – Difficult 20 points__

NO POINTS WILL BE GRANTED IF PROGRAM DOES NOT RUN DUE TO COMPILE ERRORS

Implementation: 10 points
DO NOT USE PUBLIC CLASS MEMBER VARIABLES
Determine which type of constructor is needed.
What time of functions are get_seconds, get_hours, and get_minutes?
When a user of the class calls the public function get_time 13:10:09 is returned as a string.

Test Cases: 5 points
Write test case test clock class with value 1570846218:
get_time class function returns 02:10:18 in string format.

get hours returns 2
get minutes returns 10

get seconds returns 18

Main program flow: 5 points

No loop. With value constructor argument 1570846218 create an instance of the Clock class. Call class function get_time to return and display the time.

Sample output:

```
Time: 15:45:09
```

Other example:

```
Time: 09:08:05
```