**CSC122**

**Group Project#1**

**Due 06/22/2025 by 11:59 pm**

**250 Points**

# *Exercise #1:*

Create a class called Employee that includes three private data members: a first name (type string), a last name (type string), and monthly salary (type int). The class will have a constructor that initializes the three data members. Include the set and get member functions for each data member. If the monthly salary is not positive, the set function will set it to 0. Also include a member function that returns the yearly salary (i.e. twelve times the monthly salary).

Write a test program that prompts the user to enter the first and last name of an employee as well as their monthly salary. The program will then output the yearly salary of the employee, give them a 10 percent raise, and output the new yearly salary.

Your code should be in **multiple files** with employee.h, empImp.cpp and empMain.cpp. Be sure to include the run result in an output file.

**SAMPLE RUN:** (inputs are in bold and outputs are in blue)

Enter:·the·employee's·first·and·last·name:·**John·Smith↵**

Enter:·the·employee's·monthly·salary:·**10000↵**

John·Smith's·yearly·salary·before·the·raise·was·120000↵

John·Smith's·yearly·salary·after·the·raise·is·132000↵

# *Exercise #2:*

Write a class named Car that has the following private member variables:

**year:** An int that holds the car’s model year.

**make:**   A string object that holds the make of the car.

**speed:**  An int that holds the car’s current speed.

In addition, the class should have the following public member functions.

**Constructor:** The constructor should accept the car’s year and make as arguments and assign these values to the object’s year and make member variables. The constructor should initialize the speed member variable to 0.

**Accessors:** Appropriate accessor functions should be created to allow values to be retrieved from an object’s year, make, and speed member variables.

**accelerate:** The accelerate function should add 5 to the speed member variable each time it is called.

**brake:**  The brake function should subtract 5 from the speed member variable each time it is called.

Demonstrate the class in a program that creates a Car object and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function five times. After each call to the brake function, get the current speed of the car and display it.

Your code should be in **multiple files** with car.h, carImp.cpp and carMain.cpp. Be sure to include the run result in an output file.

# *Exercise #3:*

Write the following functions, and test them all in a single main program. Do not change the given function headers or function calls.

1. Write a function

*vector append(vector a, vector b)*

that appends one vector after another.

For example, if a is

1 4 9 16

and b is

9 7 4 9 11

then append returns the vector

1 4 9 16 9 7 4 9 11

1. Write a function

*vector merge(vector a, vector b)*

that merges two vectors, alternating elements from both vectors. If one vector is shorter than the other, then alternate as long as you can and then append the remaining elements from the longer vector.

For example, if a is

1 4 9 16

and b is

9 7 4 9 11

then merge returns the vector

1 9 4 7 9 4 16 9 11

1. Write a function

*vector merge\_sorted(vector a, vector b)*

that merges two *sorted* vectors, producing a new sorted vector. Keep an index into each vector, indicating how much of it has been processed already. Each time, append the smallest unprocessed element from either vector, then advance the index.

For example, if a is

1 4 9 16

and b is

4 7 9 9 11

then merge\_sorted returns the vector

1 4 4 7 9 9 9 11 16

# *Exercise #4:*

Write the following functions, and test them all in a single main program. Do not change the given function headers or function calls. This exercise is all about pointers.

1. Write a function

void sort2(double\* p, double\* q)

that receives two pointers and sorts the values to which they point. If you call

sort2(&x, &y)

then x <= y after the call.

1. Write a function

double replace\_if\_greater(double\* p, double x)

that replaces the value to which p points with x if x is greater. Return the old value to which p pointed.

1. Write a function that computes the average value of an array of floating-point data:

double average(double\* a, int size)

In the function, use a pointer variable, not an integer index, to traverse the array elements.

1. Write a function that returns a pointer to the maximum value of an array of floating-point data:

double\* maximum(double\* a, int size)

If size is 0, return nullptr.

1. Write a function that returns a pointer to the first occurrence of the character c in the string s, or nullptr if there is no match.

char\* find(char s[], char c)

1. Write a function that returns a pointer to the last occurrence of the character c in the string s, or nullptr if there is no match.

char\* find\_last(char s[], char c)

1. Write a function that returns a pointer to the nth occurrence of the character c in the string s, or nullptr if there is no match.

char\* find\_last(char s[], char c, int n)

1. Write a function that returns a pointer to the first occurrence of the substring t in the string s, or nullptr if there is no match.

char\* find(char s[], char t[])

1. Write a function that, given strings s, t, and u, returns a string (allocated with the new operator) in which all occurrences of t in s are replaced with u:

char\* replace\_all(const char s[], const char t[], const char u[])

1. Write a function that reverses the values of an array of floating-point data:

void reverse(double\* a, int size)

In the function, use two pointer variables, not integer indexes, to traverse the array elements.

# *Exercise #5:*

Write a program that reads and prints a joke and its punch line from two different files. The first file contains a joke, but not its punch line. The second file has the punch line as its last line, preceded by “garbage.” The main function of your program should open the two files then call two functions, passing each one the file it needs. The first function should read and display each line in the file it is passed (the joke file). The second function should display only the last line of the file it is passed (the punch line file). It should find this line by seeking to the end of the file and then backing up to the beginning of the last line. Data to test your program can be found in the joke.txt and punchline.txt files.

Both files are attached to the project assignment.