**CSC 1101 – Problem Solving and Programming Laboratory**

**Lab 14 – rory lange**

**25 points – Due March 22, 11pm**

**a)** Save this document with your name and the lab assignment number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit the following documents to the Canvas assignment link where you downloaded this document:

✓ This document.

✓ Your .cpp files renamed to .txt.

Submit the documents separately, not as one .zip file.

**1) [12 points]** You’ve been hired by *Feathery Finales* to write a C++ console application that simulates the landing parameters of a NASA spacecraft. Two of the parameters we can control:

● Distance in meters from the planet when we begin slowing the spacecraft down.

● Deceleration rate at which the spacecraft slows down (this can be 0.09 or 0.95).

Two of the parameters we cannot control:

● The safe landing speed (less than or equal to 2 meters per minute.

● The spacecraft speed before slowing down (200 meters per minute).

Print three column headers:

● Minute

● Distance

● Speed

Use a while loop to run the simulation. Continue to loop as long as the distance is greater than zero. Within the loop, increment the minute counter, update the distance (distance – speed), and update the speed (speed x deceleration rate). Use formatted output manipulators (setw, left/right) to print in one row the following:

● Minute

● Distance

● Speed

After the loop, test whether the current speed is less than or equal to the safe landing speed. If not, print a crash message. If so, print a safe landing message. Define constants for the deceleration rate and the column widths. Format all real numbers to ten decimal places. Run two simulations:

● Set the deceleration rate to 0.9 and initial distance to 1,500 meters. Repeatedly run the simulation by adjusting the distance until there is no crash.

● Set the deceleration rate to 0.95 and initial distance to 1,500 meters. Repeatedly run the simulation by adjusting the distance until there is no crash.

All parameters may be set in the variable declarations. The output should look like this:

Welcome to Feathery Finales

---------------------------

Minute Distance Speed

1 1300.0000000000 180.0000000000

2 1120.0000000000 162.0000000000

3 958.0000000000 145.8000000000

4 812.2000000000 131.2200000000

5 680.9800000000 118.0980000000

6 562.8820000000 106.2882000000

7 456.5938000000 95.6593800000

8 360.9344200000 86.0934420000

9 274.8409780000 77.4840978000

10 197.3568802000 69.7356880200

11 127.6211921800 62.7621192180

12 64.8590729620 56.4859072962

13 8.3731656658 50.8373165666

14 -42.4641509008 45.7535849099

You crashed!

End of Feathery Finales

//==========================================================

//

// Title: Feathery Finales

// Course: CSC 1101

// Lab Number: 14-1

// Author: rory lange

// Date: 3/11/21

// Description:

// This C++ console application simulates the landing

// parameters of a NASA spacecraft. Two of the parameters

// we can control:

// -Distance in meters from the planet when we begin

// slowing the spacecraft down.

// -Deceleration rate at which the spacecraft slows down

// (this can be 0.09 or 0.95).

// Two of the parameters we cannot control:

// -The safe landing speed (less than or equal to 2 meters

// per minute.

// -The spacecraft speed before slowing down (200 meters

// per minute).

// It loops to run the simulation. It continues to loop as

// long as the distance is greater than zero. Within the

// loop, the appliation updates the distance (distance –

// speed) and speed (speed x deceleration rate). It then

// print in one row the following:

// -Minute

// -Distance

// -Speed

// After the loop, it tests whether the current speed is

// less than or equal to the safe landing speed. If not, it

// prints a crash message. If so, it prints a safe landing

// message.

//

//==========================================================

#include <cstdlib> // For several general-purpose functions

#include <fstream> // For file handling

#include <iomanip> // For formatted output

#include <iostream> // For cin, cout, and system

#include <string> // For string data type

using namespace std; // So "std::cout" may be abbreviated to "cout"

void SafeSpeed(double speed, const double SafeLandingSpeed = 2) {

//test whether the curent speed is less than or equal to the safe landing speed

//if not print a crash message, if so print a safe landing message

//constants for decel and column widths

//all real number s to ten decimal places

if (speed > SafeLandingSpeed)

cout << endl << "You crashed!" << endl;

else

cout << endl <<"You landed safely!" << endl;

}

int main()

{

//define variables and constants

const double Deceleration = .9;

const int w = 20;

const int w1 = 10;

double speed = 200;

double distance = 1990;

int minute = 1;

cout << fixed << setprecision(10);

//header

cout << "Welcome to Feathery Finales" << endl;

cout << "---------------------------" << endl << endl;

cout << setw(w1) << right << "Minute" << setw(w) << right << "Distance" << setw(w) << right << "Speed" << endl;

while (distance > 0) {

distance = distance - speed;

speed = speed \* Deceleration;

cout << setw(w1) << right << minute << setw(w) << right << distance << setw(w) << right << speed << endl;

minute++;

}

//say whether the ship crashes or lands safely

SafeSpeed(speed);

//app close

cout << endl << "End of calculations for Feathery Finales" << endl;

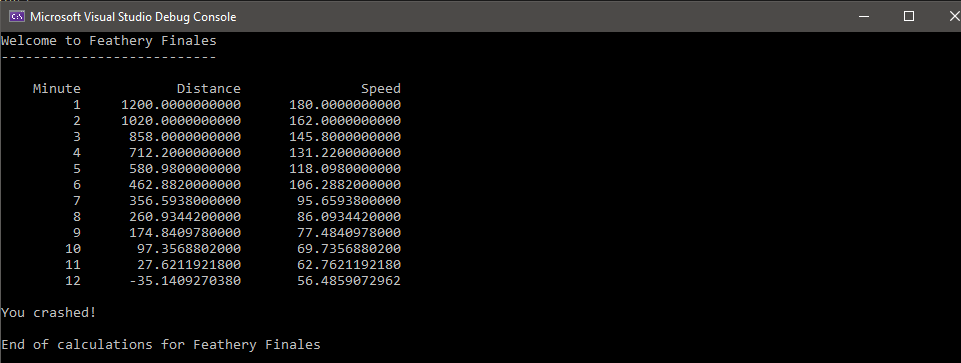
}

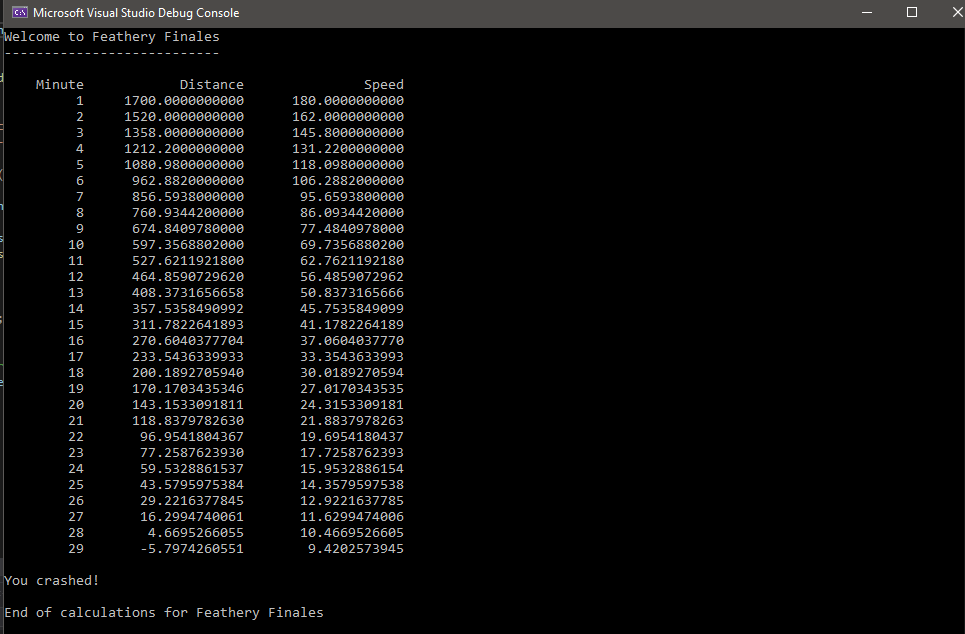
**If possible, format your code like this:**

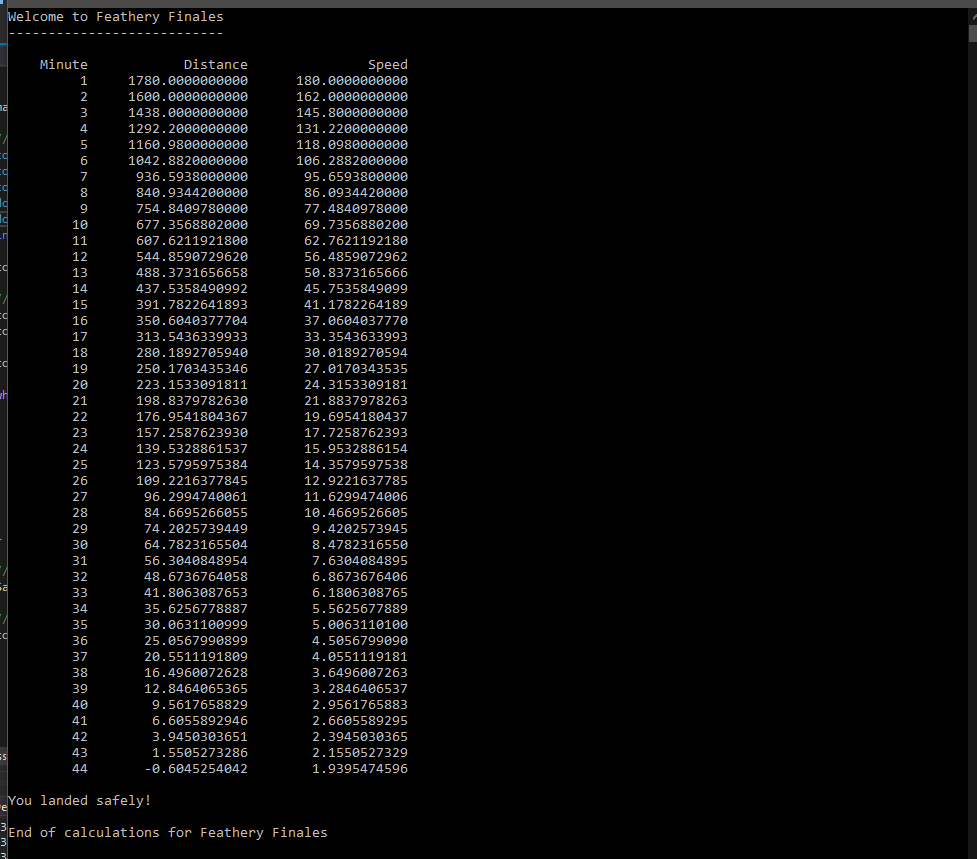
**Font “Courier New”**

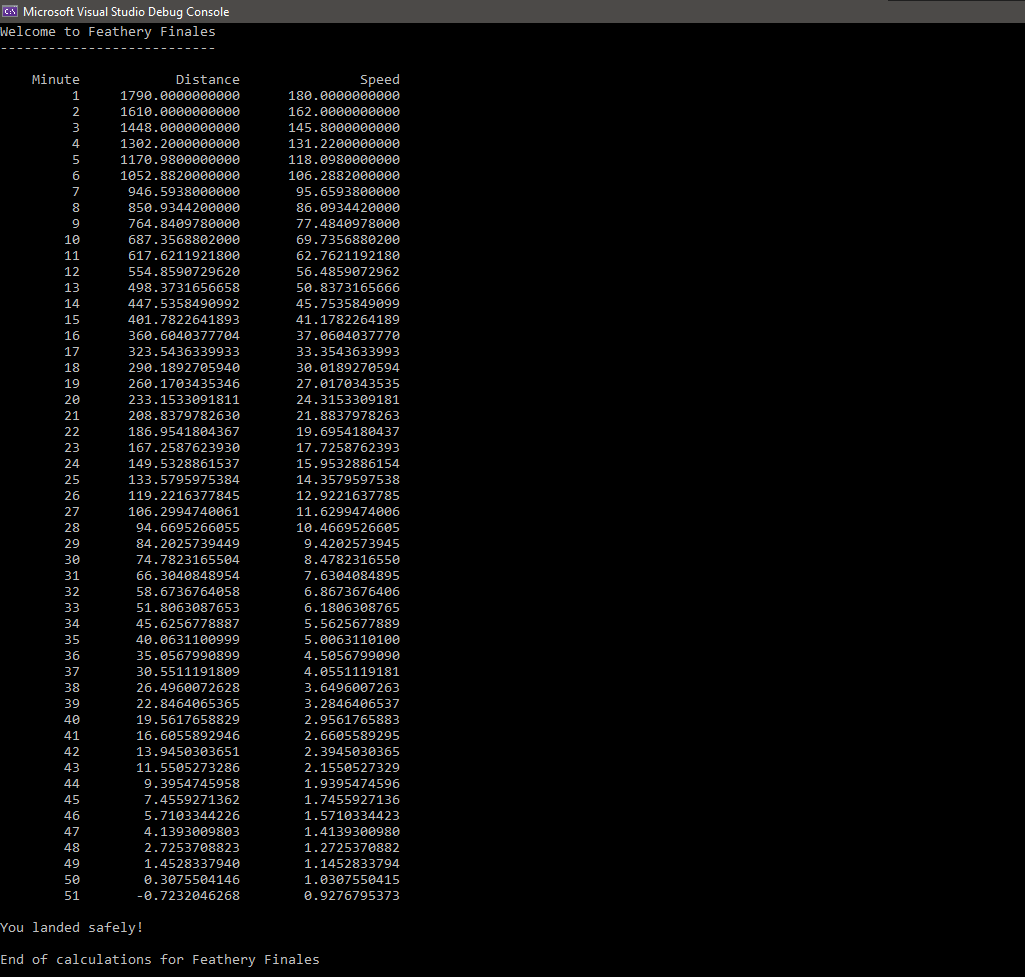
**Font size “9”**

**Bold**

**







**2) [13 points]** You've been hired by *Hidden Halos* to write a C++ console application that encodes and decodes text. Prompt for and get from the user a string that may contain any characters including spaces. Start with an empty encoded string. Encode each character of the input string per the following algorithm and append the encoded character to the encoded string. To encode each character of the input string:

● Cast the character to an integer.

● Add 57 to the integer value.

● Mod the integer value by 256 to insure its stays within the character range 0-255.

● Cast the integer back to a character.

Now, decode the encoded string. To decode each character of the encoded string:

● Cast the character to an integer.

● Subtract 57 from the integer value.

● Mod the integer value by 256 to insure its stays within the character range 0-255.

● Cast the integer back to a character.

The original string and decoded string should be identical. Use formatted output manipulators (setw, left/right) to print the following rows:

● Input string.

● The encoded string (some characters may be unprintable).

● The decoded string.

And columns:

● A left-justified label

● A left-justified value

Define constants for the shift value (57), max character value (256), and the column widths. Continue to prompt the user for strings until they enter a sentinel value of "z". Maintain a count of the number of encodings (loops) performed and print it before the application close. The output should look like this:

Welcome to Hidden Halos

-------------------------

Enter a string to encode (z to exit): Hello World!

String: Hello World!

Encoded string: uÆÖÖ£Mä£ƒÖæN

Decoded string: Hello World!

Enter a string to encode (z to exit): What's next?

String: What's next?

Encoded string: äòÄíTáM¢ÆÑíl

Decoded string: What's next?

Enter a string to encode (z to exit): z

Encodings performed:2

End of Hidden Halos

Loop the program five times with input strings different from the

sample input for the final runs pasted below.

//==========================================================

//

// Title: Hidden Halos

// Course: CSC 1101

// Lab Number: 14-2

// Author: rory lange

// Date: 3/13/21

// Description:

// This C++ console application encodes and decodes text.

// It prompts for and gets from the user a string. It then

// encodes the string by:

// -Converting each character to an integer.

// -Adding 57 to the integer value.

// -Mod'ing the integer value by 256.

// It then decodes the string, and prints the:

// -Input string

// -The encoded string

// -The decoded string

// The application continues to prompt the user for input

// strings until sentinel value 'z' is entered.

//

//==========================================================

#include <cstdlib> // For several general-purpose functions

#include <fstream> // For file handling

#include <iomanip> // For formatted output

#include <iostream> // For cin, cout, and system

#include <string> // For string data type

using namespace std; // So "std::cout" may be abbreviated to "cout"

//==========================================================

// Global constants and variables

//==========================================================

const int W = 25;

const int Shift = 57;

const int MaxValue = 256;

//==========================================================

// encode

//==========================================================

string encode(string message) {

//get char from string

//cast char to int

//add 57

//mod by 256

//cast int back to char

int a;

char end;

string encode;

for (int i = 0; i < message.length(); i++) {

a = (int) message[i];

a = a + Shift;

a = a % MaxValue;

end = (char) a;

encode = encode + end;

}

return encode;

}

//==========================================================

// decode

//==========================================================

string decode(string message) {

int a;

char end;

string decoded;

for (int i = 0; i < message.length(); i++) {

a = (int)message[i];

a = a - Shift;

a = a % MaxValue;

end = (char)a;

decoded = decoded + end;

}

return decoded;

}

//==========================================================

// main

//==========================================================

int main()

{

//declare local variables and constants

string input;

int counter = 0;

//header

cout << "Welcome to Hidden Halos" << endl;

cout << "-----------------------" << endl << endl;

while (input != "z") {

//get user input

cout << "Enter a string to be encoded (z to exit): ";

getline(cin, input);

if (input == "z") {

cout << endl;

break;

}

cout << endl << left << setw(W) << "String: " << left << setw(W) << input << endl;

cout << left << setw(W) << "Encoded String: " << left << setw(W) << encode(input) << endl;

cout << left << setw(W) << "Decoded String: " << left << setw(W) << decode(encode(input)) << endl << endl;

counter++;

}

cout << "Encodings performed: " << counter << endl << endl;

//close

cout << "End of calculations for Hidden Halos" << endl;

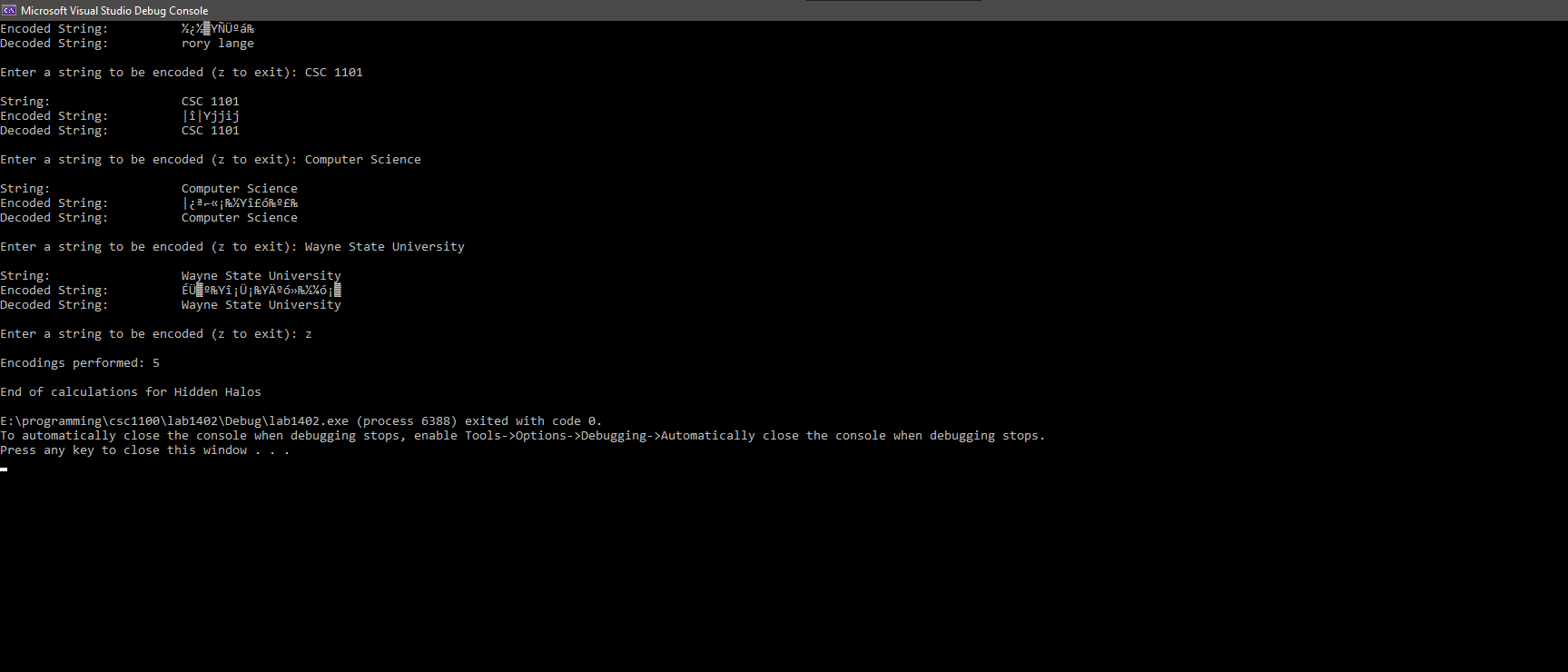
}

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”s**

**Bold**



**\* Copying-and-pasting C++ code to a Word document**

**macOS**

1) From within the C++ program, press **command-A** and press **command-C**.

2) From within the Word document, press **command-V**.

**Windows**

1) From within the C++ program, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

**\*\* Copying-and-pasting C++ console application output to a Word document**

**macOS**

1) From the C++ console, press **shift-command-4-space**.

2) From within the Word document, **command-V**.

**Windows**

1) From the C++ console, press **ALT-PrintScreen**.

2) From within the Word document, press **CTRL-V**.