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

**Lab 16 – rory lange**

**25 points – Due March 29, 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.

You've been hired by *SKU Sentries* to write a C++ console application that generates stock keeping units (SKUs) for selected car data. A SKU is an alphanumeric code created by retailers to track their products. Unlike other codes, a SKU may be any length or format. The car data comes from included file **CarData.txt** which contains a header row and 5K detail rows:

ID Name Year Seats Fuel Transmission

0001 Maruti Swift Dzire VDI 2014 5 Diesel Manual

0002 Skoda Rapid 1.5 TDI Ambition 2014 5 Diesel Manual

0006 Hyundai Xcent 1.2 VTVT E Plus 2017 5 Gas Manual

0009 Toyota Etios VXD 2011 5 Diesel Manual

0010 Ford Figo Diesel Celebration Edition 2013 5 Diesel Manual

…

Here is the specification for the file:

Field Type Start End

ID string 0 9

Name string 10 65

Year string 66 75

Seats string 76 82

Fuel string 83 94

Transmission string 95 104

The application will read this file and write another file. Attempt to open the input file. If it doesn't open, print an error message and end the application. If it does open, attempt to open the output file. If it doesn't open, print an error message and end the application. If it does open, read past the header row in the input file and print a message that the application is reading from the input file and writing to the output file. Loop through the input file, read each line at a time, and update the number of lines read. For each line, use the **substr** function to parse it into its six fields: ID, name, year, seats, fuel, and transmission. If the name begins with "Ford", then generate a SKU for the car that concatenates the following parts:

"FRD-"

The last two digits of year field.

"-S" followed by the first character of seats field.

"F" followed by the first character of fuel field.

"T" followed by the first character of transmission field.

For example, for the following car:

0010 Ford Figo Diesel Celebration Edition 2013 5 Diesel Manual

The SKU will be:

FRD-13-S5FDTM

Update the number of SKUs generated. If the number generated is a multiple of 10, print the ID, SKU, and full name for the car. For every generated SKU, write ID, SKU, and full name for the car. After the read loop, print the number of lines read and written, and close the files. Define global constants for the input and output file names, and from the file specification, the spot where each field starts and the length of each field. See sample application **Text file input and output per file specification** on Canvas. The output should look like this:

Welcome to SKU Sentries

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

Reading lines from file 'CarData.txt'

and writing lines to file 'CarDataWithSKUs.txt' ...

0449 FRD-16-S5FGTM Ford Figo 1.2P Titanium Plus MT

0616 FRD-12-S5FDTM Ford Figo Diesel ZXI

…

6107 FRD-16-S5FDTM Ford Ecosport 1.5 Diesel Titanium

… line(s) read from file 'CarData.txt'.

… SKU(s) generated and written to file 'CarDataWithSKUs.txt'.

End of SKU Sentries

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

//

// Title: SKU Sentries

// Course: CSC 1101

// Lab Number: 16-1

// Author: rory lange

// Date: 3/26/21

// Description:

// This C++ console application generates stock keeping

// units (SKUs) for selected car data. A SKU is an

// alphanumeric code created by retailers to track their

// products. Unlike other codes, a SKU may be any length

// or format. The car data comes from included file

// CarData.txt which contains a header row and 5K detail

// rows. Here is the specification for the file:

//

// Field Type Start End

// ID string 0 9

// Name string 10 65

// Year string 66 75

// Seats string 76 82

// Fuel string 83 94

// Transmission string 95 104

//

// The application reads this file and writes another file.

// It attempts to open the input file. If it doesn't open,

// it prints an error message and ends the application. If

// it does open, it attempts to open the output file. If it

// doesn't open, the application prints an error message and

// ends the application. If it does open, it reads past the

// header row in the input file and prints a message that the

// application is reading from the input file and writing to

// the output file. The application loops through the input

// file, reads each line at a time, and updates the number

// of lines read. For each line, it uses the substr function

// to parse it into its six fields: ID, name, year, seats,

// fuel, and transmission. If the name begins with "Ford",

// it generates a SKU for the car that concatenates the

// following parts:

// "FRD-"

// The last two digits of year field.

// "-S" followed by the first character of seats field.

// "F" followed by the first character of fuel field.

// "T" followed by the first character of transmission field.

//

// It updates the number of SKUs generated. If the number

// generated is a multiple of 10, it prints the ID, SKU, and

// full name for the car. For every generated SKU, the

// application writes the ID, SKU, and full name for the

// car. After the read loop, it prints the number of lines

// read and written, and closes the files.

//

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

#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"

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

// Globals constants

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

const string ReadFileName = "CarData.txt";

const string WriteFileName = "CarDataWithSKUs.txt";

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

// create SKU

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

string make\_SKU (string year, string seats, string fuel, string trans, string name = "FRD") {

string SKU;

SKU = name + "-" + year + "-S" + seats + "F" + fuel + "T" + trans;

return SKU;

}

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

// main

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

int main()

{

//header

cout << "Welcome to SKU Sentries" << endl;

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

ofstream outputFile;

ifstream inputFile;

outputFile.open(WriteFileName);

inputFile.open(ReadFileName);

string line;

string id;

string year;

string seats;

string fuel;

string trans;

string SKU;

int lineCount = 2;

int skuCount = 1;

if (!inputFile.is\_open())

cout << "input not working" << endl;

if (!outputFile.is\_open())

cout << "output not working" << endl;

else {

getline(inputFile, line);

outputFile << left << setw(10) << line.substr(0, 5) << left << setw(15) << "SKU" << line.substr(10) << endl;

cout << "Reading lines from file " << ReadFileName << endl;

cout << "Writing lines to file " << WriteFileName << endl << endl;

while (inputFile.good() && outputFile.good()) {

getline(inputFile, line);

if (line.substr(10, 4) == "Ford") {

skuCount++;

id = line.substr(0, 4);

year = line.substr(68, 2);

seats = line.substr(76, 1);

fuel = line.substr(83, 1);

trans = line.substr(95, 1);

SKU = make\_SKU(year, seats, fuel, trans);

cout << left << setw(17) << SKU << line.substr(10, 55) << endl;

if (skuCount % 10 == 0)

outputFile << left << setw(10) << id << left << setw(15) << SKU << left << line.substr(10) << endl;

}

lineCount++;

}

cout << endl << lineCount << " lines read." << endl;

cout << skuCount << " SKUs created." << endl << endl;

//ending

cout << "End of SKU Sentries." << endl;

}

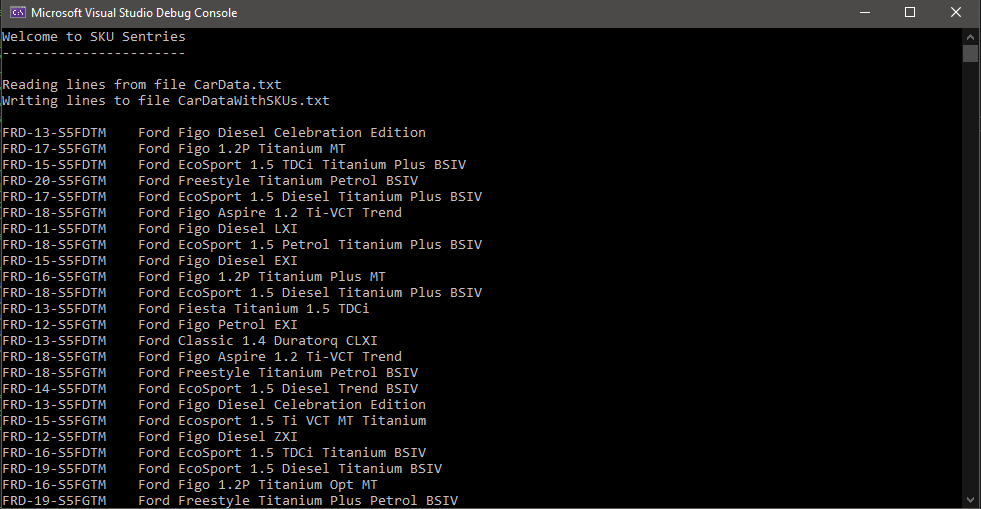
}

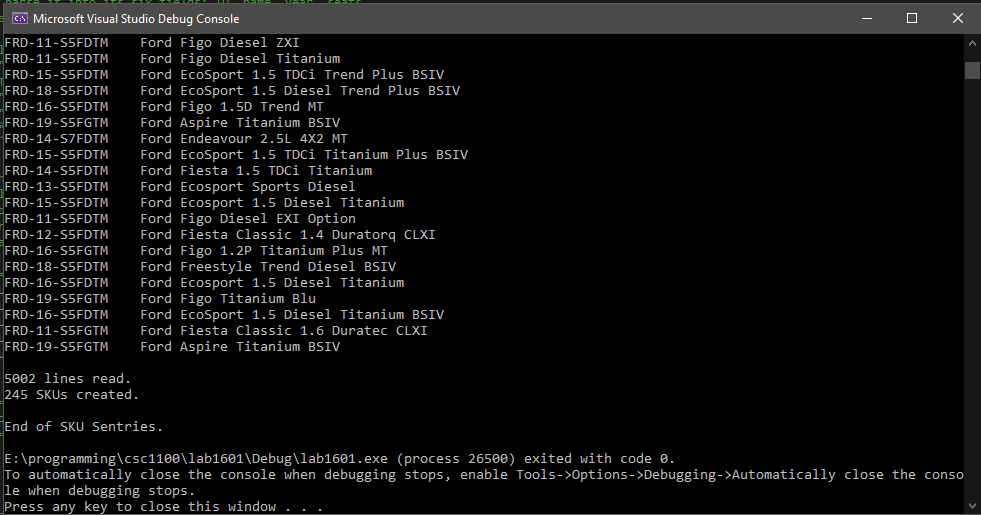
**If possible, format your code like this:**

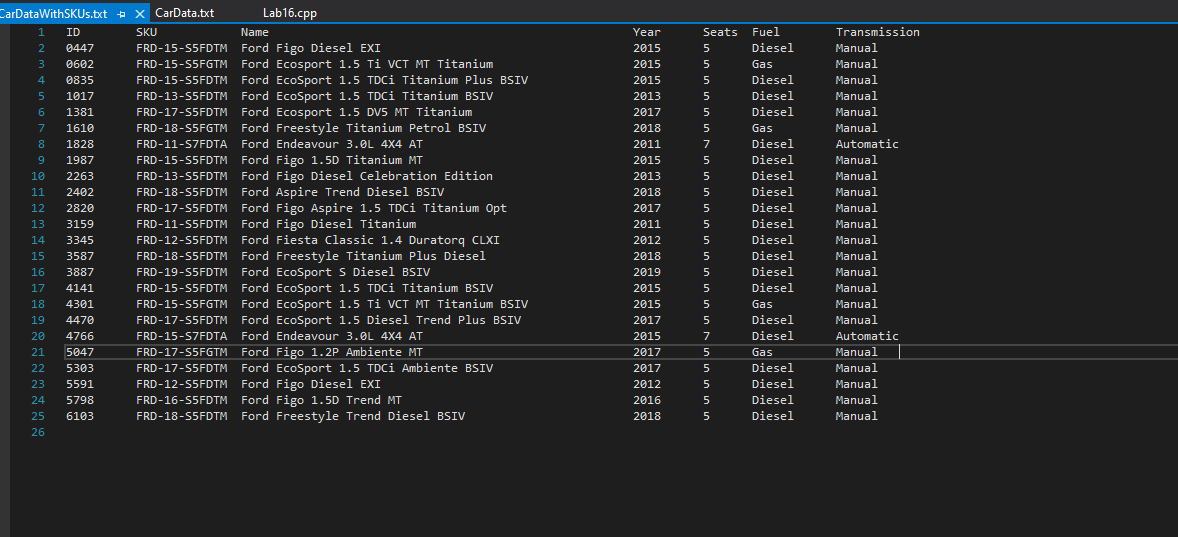
**Font “Courier New”**

**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**.