# CSC242 Introduction to Programming Concepts Week Three Assignment

| **First Name** | Robert |
| --- | --- |
| **Last Name** | Dykes |
| **ID#** | 041502945 |
| **Email Address** | Rjdykes1010805@outlook.com |

Table of Contents

[How to submit your Assignment 2](#_Toc535167619)

[P1 – Temperature Conversions 2](#_Toc535167620)

[P2 – Displaying a Square of Asterisks 3](#_Toc535167621)

[P3 – Perfect Numbers 3](#_Toc535167622)

[P4 – Assign Grades 4](#_Toc535167623)

[P5 – Pattern Recognition: Consecutive Four Equal Numbers 4](#_Toc535167624)

[P6 – Common Elements 5](#_Toc535167625)

# 

# How to submit your Assignment

After filling all the parts in this file, please follow the following steps.

1. Add your name and ID to the first page.
2. Save the file in the original format (Docx or Doc)

(please do not convert to other file formats e.g. PDF, ZIP, RAR, …).

1. Rename the file as

CSC242 *–* ***HW3*** *- ID – YOUR Last Name - YOUR First Name.docx*

**Example:** CSC242 *–* ***HW3*** *-* 234566435 - Smith - John.docx

1. Upload the file and submit it (only using Blackboard)

# P1 – Temperature Conversions

Implement the following integer functions:

**a)** Function ***celsius*** returns the Celsius equivalent of a Fahrenheit temperature, using the calculation

*celsius = 5.0 / 9.0 \* (fahrenheit - 32);*

**b)** Function ***fahrenheit*** returns the Fahrenheit equivalent of a Celsius temperature, using the calculation

*fahrenheit = 9.0 / 5.0 \* celsius + 32;*

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  using std::cout;  using std::cin;  using std::endl;  using std::string;  double celsius(double num) {  double celsius = 5.0 / 9.0 \* (num - 32);  return celsius;  };  double fahrenheit(double num) {  double fahrenheit = 9.0 / 5.0 \* (num + 32);  return fahrenheit;  }  int main() {  string temp;  double num;  string str;  cout << "Please enter f or c to convert to Fahrenheit and Celsius respectively, followed by a space and then the number to convert and press enter: " << endl;  cin >> temp >> num;    if (temp == "f" && temp.length() == 1) {  /\* ran into a logical error that would make the program hang if entered (temp.length - 1 == 0) if multiple characters were in temp, however, A normal invalid input was reached if a value for num is entered.... curious. I'm assuming the value of 0 equating to false being the culprit \*/  cout << fahrenheit(num) << endl;  cout << temp.length() - 1 << endl;  } else if (temp == "c" && temp.length() == 1) {  cout << celsius(num) << endl; }  else {  cout << "Invalid input, try again" << endl;  }    return 0;  } |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

# 

# P2 – Displaying a Square of Asterisks

Write a function **squareOfAsterisks** that displays a solid square (the same number of rows and columns) of asterisks whose side is specified in integer parameter **side**. For example, if **side** is 4, the method should display

|  |
| --- |
| \*\*\*\*  \*\*\*\*  \*\*\*\*  \*\*\*\* |

In your **main()** function, read an integer value function for side from the user and call the **squareOfAsterisks** and display the asterisks.

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  using std::endl;  using std::cin;  using std::cout;  void squareOfAstrisks(int num) {  for (int i = 0; i < num; i++) {  for (int j = 0; j < num; j++) {  cout << "\* ";  }  cout << endl;  }  }  int main() {  int num;  cout << "Please enter a number for ye ol' rough and tumble box o' astrisks" << endl;  cin >> num;  // curious why I get an error when trying cout squareOfAstrisks as a void function  // I dont want to return anything  // takeing out cout was the trick and just calling the function works, which makes sense... can't cout a cout i guess.  // cout << squareofStrisks(num) << endl;  squareOfAstrisks(num);  } |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

# P3 – Perfect Numbers

An integer number is said to be a perfect number if its factors, including 1 (but not the number itself), sum to the number.

For example, **6** is a perfect number, because **6 = 1 + 2 + 3**.

Write a function **isPerfect** that determines whether parameter number is a perfect number.

Use this method in an program that displays all the perfect numbers between 1 and 1,000.

Display the factors of each perfect number to confirm that the number is indeed perfect.

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  using std::endl;  using std::cin;  using std::cout;  bool isFactor(int factor, int whole) {  if (whole % factor == 0) {  return true;  } else {  return false;  }  }  bool isPerfect(int num) {  int temp = 0;  for (int i = 1; i <= num / 2; i++) {  if (isFactor(i, num)) {  temp += i;  }  }  if (temp == num) {  return true;  } else {  return false;  }  }  void displayFactors(int num) {  for (int i = 1; i <= num / 2; i++) {  if (num % i == 0) {  cout << "A factor of " << num << " is: " << i << endl;  }  }  }  /\* I think displayFactors and isperfect could be optimized for time complexity? it seems the instructions wanted isPerfect to strictly return a boolean maybe I'm missing something? \*/  int main() {  for (int i = 0; i < 1000; i ++) {  if (isPerfect(i)) {  cout << i << " is a perfect number!" << endl;  displayFactors(i);  }  }  return 0;  } |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

# P4 – Assign Grades

Using an array of size 100. write a program that reads student scores, gets the best score, and then assigns grades based on the following scheme:

Grade is A if score is 7 = best - 10;

Grade is B if score is 7 = best - 20;

Grade is C if score is 7 = best - 30;

Grade is D if score is 7 = best - 40;

Grade is F otherwise.

The program prompts the user to enter the total number of students, then prompts the user to enter all of the scores, and concludes by displaying the grades. Here is a sample run:

|  |
| --- |
| Enter the number of students: 4  Enter 4 scores: 40 55 70 58  Student 0 score is 40 and grade is C  Student 1 score is 55 and grade is B  Student 2 score is 70 and grade is A  Student 3 score is 58 and grade is B |

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  using std:: endl;  using std:: cin;  using std:: cout;  int main() {  int students;  int scores[100];  cout << "Please enter the number of students and press enter" << endl;  cin >> students;  cout << "Please enter scores for each student seperated by space and press enter" << endl;  for (int i = 0; i < students; i++) {  cin >> scores[i];  }  int bestStudent = 0;  for (int i = 0; i < students; i++) {  if (scores[i] > bestStudent) {  bestStudent = scores[i];  }  }    for (int i = 0; i < students; i++) {  if (scores[i] >= 0 && scores[i] <= 100) {  if (scores[i] >= bestStudent - 10) {  cout << "Student " << i << " recieves an A!" << endl;  continue;  }  if (scores[i] >= bestStudent - 20) {  cout << "Student " << i << " recieves an B." << endl;  continue;  }  if (scores[i] >= bestStudent - 30) {  cout << "Student " << i << " recieves an C." << endl;  continue;  }  if (scores[i] >= bestStudent - 40) {  cout << "Student " << i << " recieves an D." << endl;  continue;  } else {  cout << "Student " << i << " recieves an E?" << endl;  continue;  }  } else {  cout << "Please start over with valid inputs" << endl;  break;  }  }  return 0;  } |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

# P5 – Pattern Recognition: Consecutive Four Equal Numbers

Write the following function that tests whether the array has four consecutive numbers with the same value.

**bool** isConsecutiveFour(**const** **int** values[], **int** size)

Write a test program that prompts the user to enter a series of integers and displays if the series contains four consecutive numbers with the same value.

Your program should first prompt the user to enter the input size—i.e., the number of values in the series. Assume the maximum number of values is 80.

Here are sample runs:

|  |
| --- |
| Enter the number of values: 8  Enter the values: 3 4 5 5 5 5 4 5  The list has consecutive fours |

|  |
| --- |
| Enter the number of values: 9  Enter the values: 3 4 5 5 6 5 5 4 5  The list has no consecutive fours |

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  using std::cin;  using std::cout;  using std::endl;  int main() {  int values;  int valueArray[30];  int valueCount = 1;  int value = 0;  bool firstTime = true;  bool consecutiveValues = false;  cout << "Please enter how many values you wish to enter and press enter" << endl;  cin >> values;  cout << "Please enter the specific values seperated by space and press enter" << endl;  for (int i = 0; i < values; i++) {  cin >> valueArray[i];  }  for (int i = 0; i < values; i++) {  if (firstTime) {  firstTime = false;  value = valueArray[i];  continue;  }  if (valueArray[i] == value) {  valueCount++;  }  if (valueCount == 4) {  cout << "There are atleast four consecutive values!" << endl;  consecutiveValues = true;  break;  }  if (valueArray[i] != value) {  valueCount = 1;  value = valueArray[i];  continue;  }  }  if (consecutiveValues == false) {  cout << "There are not aleast 4 consecutive values" << endl;  }  return 0;  } |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

# P6 – Common Elements

Write a program that prompts the user to enter two arrays of 10 integers and displays the common elements that appear in both arrays. Here is a sample run.

|  |
| --- |
| Enter list1: 8 5 10 1 6 16 61 9 11 2  Enter list2: 4 2 3 10 3 34 35 67 3 1  The common elements are 10 1 2 |

|  |
| --- |
| Your C++ code for this problem |
| #include <iostream>  #include <string>  using std::cin;  using std::cout;  using std::endl;  using std::string;  using std::to\_string;  int main() {  int firstArray[10];  int secondArray[10];  int usedNumsArray[100];  string display;  bool anyDuplicates = false;  cout << "Please enter 10 numbers between 1 and 100" << endl;  for (int i = 0; i < 10; i++) {  cin >> firstArray[i];  }  cout << "Please enter 10 more numbers to compare with first 10 between 1 and 100" << endl;  for (int i = 0; i < 10; i++) {  cin >> secondArray[i];  }  for (int i = 0; i < 10; i++) {  for (int j = 0; j < 10; j++) {  if (secondArray[j] == firstArray[i] && usedNumsArray[secondArray[j]] != secondArray[j]) {  anyDuplicates = true;  usedNumsArray[secondArray[j]] = secondArray[j];  display += to\_string(secondArray[j]);  display += " ";  }  }  }    if (anyDuplicates) {  cout << "The duplicates are: " << endl;  cout << display << endl;  } else {  cout << "There are no duplicates to display" << endl;  }  return 0;  }  /\* i'm sure there is a better way of ensuring that no duplicates fall through the cracks. my method here was to create a "have we seen this number before" array (usedNumsArray) and make the number that was identical between input arrays equal to the index and the number held at the index within UsedNumsArray. my previous homework I used a string "abcdefghijklmnopqrstuvwxyz" to prevent seeing used characters twice... i'm sure im missing a better practice \*/ |

Run the code and insert the result in the following box.

|  |
| --- |
| The run result |
|  |

**The end**