Question 1

#include <iostream>

#include <string>

using namespace std;

int SumOfArray(int list[], int size)

{

int sum = 0;

for (int i = 0; i < size; i++)

{

sum += list[i];

}

return sum;

}

double AverageOfPostitves(int list[], int size)

{

int avg = 0;

int numPos = 0;

for (int i = 0; i < size; i++)

{

if (list[i] >= 0)

{

avg += list[i];

numPos++;

}

}

avg /= static\_cast<double>(numPos);

return avg;

}

int Lowest(int list[], int size)

{

int lowestNum = list[0];

for (int i = 0; i < size; i++)

{

if (list[i] < lowestNum)

{

lowestNum = list[i];

}

}

return lowestNum;

}

int Highest(int list[], int size)

{

int highest = list[0];

for (int i = 0; i < size; i++)

{

if (list[i] > highest)

{

highest = list[i];

}

}

return highest;

}

int main()

{

const int SIZE = 20;

int values[SIZE] = { 0, 23, 34, -7, 110, 42, -350, 424, 25,

99, 10, 05, 50, -5, 1, 200, -350, 437, 25, 147 };

cout << "Highest number is " << Highest(values, SIZE) << endl;

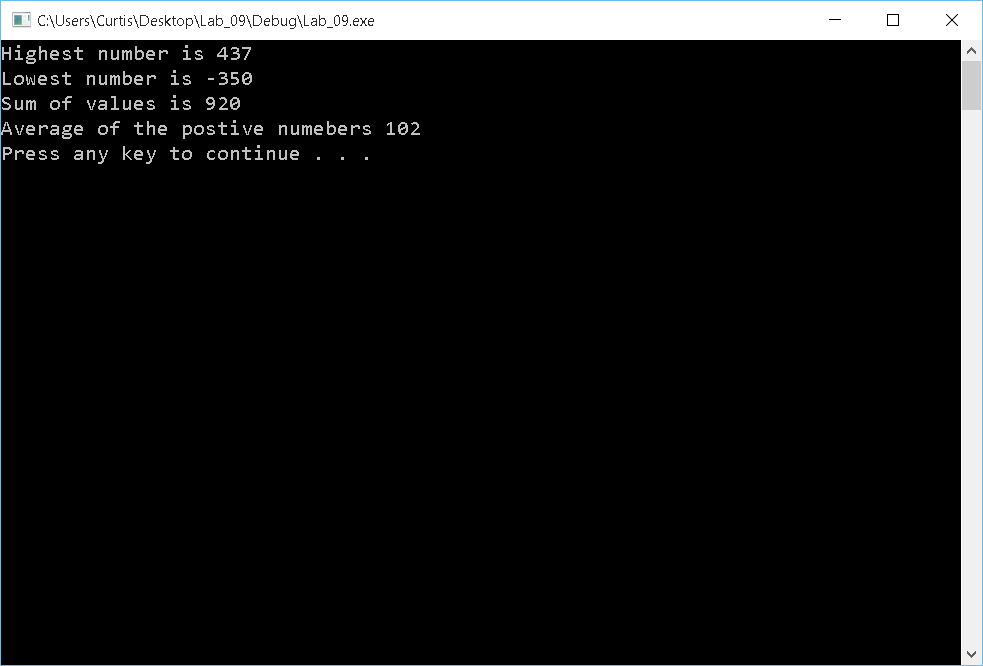
cout << "Lowest number is " << Lowest(values, SIZE) << endl;

cout << "Sum of values is " << SumOfArray(values, SIZE) << endl;

cout << "Average of the postive numebers " << AverageOfPostitves(values, SIZE) << endl;

system("pause");

}



Question 2

#include <iostream>

#include <string>

#include <fstream>

#include <cctype>

using namespace std;

// Author:

// Creation Date:

// Last Modification Date:

// Purpose: compares answer with key (character per character).

// Each character represents the answer to a question.

// The number of questions is numOfQuestions.

// Returns the number of correct answers

int gradeAnswer(string answers, string key, int numOfQuestions)

{

int numCorrect = 0;

for (int i = 0; i < numOfQuestions; i++)

{

if (toupper(answers[i]) == key[i])

{

numCorrect++;

}

}

return numCorrect;

}

// Author:

// Creation Date:

// Last Modification Date:

// Purpose: Gets the number of correct questions and the total

// number of questions. Returns the grade out of 100

float calculatePointGrade(int numOfCorrectAnswers, int

numOfQuestions)

{

return static\_cast<float> (numOfCorrectAnswers) / numOfQuestions \* 100.00;

}

// Author:

// Creation Date:

// Last Modification Date:

// Purpose: Returns the letter grade using the following:

// >=90 A

// <90 and >=80 B

// <80 and >= 70 C

// <70 and >= 60 D

// <60 and >= 50 E

// <50 F

string calculateLetterGrade(float pointGrade)

{

if (pointGrade >= 90)

{

return "A";

}

else if (pointGrade < 90 && pointGrade >= 80)

{

return "B";

}

else if (pointGrade < 80 && pointGrade >= 70)

{

return "C";

}

else if (pointGrade < 70 && pointGrade >= 60)

{

return "D";

}

else if (pointGrade < 60 && pointGrade >= 50)

{

return "E";

}

else if (pointGrade < 50)

{

return "F";

}

}

// Author:

// Creation Date:

// Last Modification Date:

// Purpose: Returns the number of questions and answer

// key of the exam (both as reference parameters)

// We assume the file settings contains only two date:

// Number of questions (in line 1 of the file)

// Answer key (line 2 of the file)

void readSettings(ifstream& settings, string& key, int&

numOfQuestions)

{

settings >> numOfQuestions;

settings >> key;

return;

}

int main()

{

int numQuestions = 0;

string key = "";

ifstream pullSet;

ifstream pullExamGrades;

ofstream push;

string firstName;

string LastName;

string Answers;

int numCorrect;

string letterGrade;

float percent;

pullSet.open("settings.txt");

readSettings(pullSet, key, numQuestions);

pullExamGrades.open("exam.txt");

push.open("grades.txt");

while (!pullExamGrades.eof())

{

pullExamGrades >> firstName;

pullExamGrades >> LastName;

pullExamGrades >> Answers;

numCorrect = gradeAnswer(Answers, key, numQuestions);

percent = calculatePointGrade(numCorrect, numQuestions);

letterGrade = calculateLetterGrade(percent);

push << firstName << " " << LastName << " " << percent << "% " << letterGrade << endl;

cout << firstName << " " << LastName << " " << percent << "% " << letterGrade << endl;

}

system("pause");

return 0;

}

