Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Consider the function definition:**

void Twist( int a, int& b )

{

int c;

c = a + 2;

a = a \* 3;

b = c + a;

}

* 1. **Suppose that the caller has variables:**

**int r = 1, int s = 2, and int t = 3**

**What are the values of r, s and t after return from the following function call?**

Twist(t, s); // precondition, postcondition \*\*this is the format I put s and t in answer-wise

r**: 1 // this should not be affected by the function as it is not called to it nor is it defined in within its scope**

**c is initialized to 4 when c = a(s) + 2;**

s**: 2, 6 // called by reference**

t: 3, 8 // called by value \*\*new data location

* 1. **Also, describe what happens in MEMORY when the Twist function is called. Think about the by reference, local variables and by value parameters.**

The value represented by the variable t is passed through to the Twist function and a copy variable is created and stored at different memory location with the same value stored, however, the value stored at the memory location allocated for int b (or s in this case) is accessed in Twist via a passing of the location of the original variable. So, where t (or a) is passed through a copied value, s (or b) has its location sent to the Twist function so that the function may use the same memory location as the functions whose parameters encapsulate Twist. R is not interacted with by the Twist function since neither when the function is called nor when the variables within it are declared is there any mention of the variable ‘r’.

1. **For the following switch statement;**

**a. What is the output of the following code fragment if the input value is 4? (Be careful here.)**

**10+2+3+4 = 19**

**b. Suppose the code should output a single value per case, would the switch segment output the proper value(s) in that situation?**

**No**

**If NO, what statement(s) would you add to the above switch statement so the output would always be one per case statement? You do not have to rewrite the code segment, you may indicate the correction(s) in the existing code.**

int num;

int alpha = 10;

cin >> num;

switch (num)

{

case 3 : alpha++;//+break;

case 4 : alpha = alpha + 2; //+break;

case 8 : alpha = alpha + 3; //+break;

default : alpha = alpha + 4; //+break;

}

cout << alpha << endl;

1. **Write a program that reads float values from a file, sums the values and calculates the average.**
2. **Write a value-returning function that opens an input file named inFile.txt. You may "hard-code" the file name, i.e., you do not need to ask the user for the file name. The function should check the file state of the input file and return a value indicating success or failure. Main should check the returned value and if the file open fails, output a message to cout and exit the program.**
3. **Write a void function that reads float values from the data file, sums the values and calculates the average of the values. Return the sum of the values, number of values read and the average of the values to the calling module, main. Since you do not know the number of values in the file, you must check for end of file.**
4. **In main, output to cout the sum, number of values read and the average.**
5. **Include the function prototypes**
6. **Declare the variables necessary to create a running program.**
7. **Write statements for all include files, using statements and main.**

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**using namespace std;**

**int someOpeningsBetterHappen(ifstream&, ofstream&, string&,**

**string&, string&, int&);**

**void readFile(float&, ifstream&, int&, int&);**

**void closeFiles(ifstream&, ofstream&);**

**int main () {**

**ifstream fromFile;**

**ofstream toFile;**

**string inputFile, outputFile, reportFile;**

**int runThrough = 0;**

**float avg = 0.00;**

**int numValues, total = 0;**

**inputFile = "inFile.txt";**

**outputFile = "outFile.txt";**

**while(runThrough < 2){**

**someOpeningsBetterHappen(fromFile, toFile, inputFile, outputFile, reportFile, runThrough);**

**if(someOpeningsBetterHappen == 0)**

**{**

**cout << "Error opening file: " << reportFile << ". Exiting program..." << endl;**

**return 0;**

**}**

**runThrough++;**

**}**

**readFile(avg, fromFile, numValues, total);**

**cout << "You entered " << numValues << " numbers totaling " << total << " with a average of " << avg << endl;**

**toFile << "You entered " << numValues << " numbers totaling " << total << " with a average of " << avg << endl;**

**closeFiles(fromFile, toFile);**

**}**

**int someOpeningsBetterHappen(ifstream& inData, ofstream& outData, string&**

**ifileName, string& ofileName, string& fileToReport, int& instance){**

**if(instance == 0){**

**inData.open(ifileName);**

**}else if(instance == 1){**

**outData.open(ofileName);**

**}**

**if (!outData) {**

**fileToReport = ofileName;**

**return 0;**

**}else{**

**return 1;**

**}**

**}**

**void readFile(float& avg, ifstream& inData, int& numValues, int& total)**

**{**

**int num;**

**inData >> numValues;**

**for (int i = 0; i < numValues; i++)**

**{**

**inData >> num;**

**cout << num;**

**total += num;**

**}**

**avg = float(total) / float(numValues);**

**}**

**}**

**void closeFiles(ifstream& inData, ofstream& outData)**

**{**

**inData.close();**

**outData.close();**

**}v**

1. **Consider the following code segment. What is the value of loopcount** **and total** **after the for loop completes?**

int total = 0, loopcount = 0;

for (int loopcount = 0; loopcount < 5; loopcount ++){

total = total + loopcount;

}

cout << " Total is: " << total << " loopcount is: " << loopcount << endl;

**loop1: loopcount = 0, total = 0**

**loop2: loopcount = 1, total = 1**

**loop3: loopcount = 2, total = 3**

**loop4: loopcount = 3, total = 6**

**loop5: loopcount = 4, total = 10 //total is the only one that changes due to the scope in which it is declared being outside of the for loop, whereas the for loop’s scope contains that definition of loopcount, while another loopcount, declared to 0, remains in the outer scope. Thus, Loopcount == 0 and total == 10.**

0=0+0

1=0+1

3=1+2

6=3+3

10=6+4

**loopcount: 5**

**total: 10**

1. **What is the output of the following code fragment? (Be careful here.)**

count = 1;

while (count < 10)

count++;

cout << "Hello"; // Im gonna assume that the minor indentation above this line indicates

//where the loops contents are and that the “cout<<’hello’;” is outside of the //loop, and if that is the case, then the output is simply: Hello

1. **Write a For loop that is equivalent to the following While loop. (All variables are of type int.)**

count = -5;

while (count <= 15){

sum = sum + count;

count++;

}

for(int count = -5; count <= 15; count++){

sum = sum + count;

}