1. Write a function ***checkFile*** that opens up a file. The function takes one parameter, the name of the file *filename*. The function should return **true** if the file was open successfully otherwise it should return **false**.

For example, assume a file called *myFile.txt* exists in your workspace, then a function call to ***checkFile****(...)*

    checkFile("myFile.txt")

       would return **true** because the file exists in the system.

bool checkFile(string filename){

ofstream myfile;

myfile.open(filename);

if(myfile.is\_open()){

return false;

}

else

return true;

}

2. Write a function *fileLoadWrite* that opens a file and writes to it the squares of the numbers 1-10, each on its own line. The function takes one parameter, the name of the file *filename.*It should open the file in write mode and write 10 lines to the file, each containing the square of the number line it is on (e.g. line 1 will contain 1, line 2 will contain 4, etc.). The function should close the file and return 0 if the file was opened successfully, else it should return -1.

An example of a function call to *fileLoaderWrite* is fileLoadWrite("myFile.txt") which should successfully open a new file called *myFile.txt* and write ten lines of squared numbers.

Note: If a file called *myFile.txt* is already associated to a stream then the open(...) method fails

int fileLoadWrite(string filename){

ofstream myfile;

myfile.open(filename);

if(myfile.fail()){

return -1;

}

if(myfile.is\_open()){

for(int i=1; i<=10; i++){

myfile<<i\*i<<"\n";

}

}

return 0;

}

Write a function *fileLoadRead* that reads a file. The function takes a parameter, the name of the file *filename,*and returns the number of lines in the file. It should open a file in read mode, check for success, and read the lines until the end of the file. The function should close the file and return return the number of lines if the file was open successfully otherwise it should return -1.

For Example:

Given a file called *myFile.txt*with the following content:

hola  
ciao  
hello  
hallo

the function call fileLoadRead("myFile.txt") would return 4.

int fileLoadRead(string filename){

int count=0;

string line;

ifstream myfile;

myfile.open(filename);

if(myfile.is\_open()){

while(getline(myfile, line)){

count++;

}

return count;

}else

return -1;

myfile.close();

}

Write a function ***getLinesFromFile*** that reads from a file and stores its content in an array. The function takes three parameters: a string *fileName,* a string array *wordArray,* an integer *sizeArray*.

***getLinesFromFile***should open the file in read mode, read each line, store each line in the array.  The function should return the number of lines placed into the array. If the file does not exist, return -1.

Example: if *fileName.txt* has the following contents:

sky  
night  
78  
ski season

The function call

***getLinesFromFile****("fileName.txt", wordArray, 4)*

would return 4 and wordArray would look like ["sky", "night", "78", "ski season"]

int getLinesFromFile(string filename, string wordArray[], int sizeArray){

int count=0;

string line;

ifstream myfile;

myfile.open(filename);

if(myfile.is\_open()){

for(int i=0; getline(myfile, line); i++){

wordArray[i]=line;

count++;

}

return count;

}else

return -1;

myfile.close();

}

After a program finishes executing, intermediate results are lost unless we save them somewhere. Frequently, this can be accomplished by writing the results to a file. For this problem, you must write a function ***saveData***which takes three arguments: the filename, an array of strings, and the size of the array. The given array of strings has the following structure:

"value","value","value","name"

This function will convert the first 3 values of the array into floats, compute their average and write to a file the name stored in the array of strings and on the next line, the average computed in string format. If the input is not in valid format the function should not write to the file.

void saveData(string fileName, string data[], int size) {  
 //Your code here  
}

**Example Output**

string data[4] = {"2.3", "-1.5", "0.8", "Garth"};

**saveData**("my\_data.txt", data, 4);

The file "my\_data.txt" should have the following two lines:

Name: Garth  
Avg: 0.5333