Homework Turnin

Name:	Minsu Jung
Email:	minsu25100@gmail.com
Section:	A
Course:	CS 132 23sp
Assignment:	p1
Receipt ID:	1276b8c065442bf338fe3743bac9f49a

Turnin Successful!

The following file(s) were received:

madlibs.cpp (1081 bytes)

```
/*
* Minsu Jung
* CS 132 Project 1
* April 8
* This code is the main function for a game that prompts the user to ...
* select a game mode (create, view, or quit) and executes the corresponding function. ...
* It loops continuously until the user chooses to quit.
*/
#include <iostream>
#include <vector>
#include <string>
#include "lib132.cpp"

using namespace std;
int main()
{
   GameStartNotification();   // Notify the start of the game
   std::string userInput;

   // Game processing loop
   while (true)
   {
      std::cout << "Create, view or quit? ";
      std::getline(std::cin, userInput);
      int gameMode = GameModeSetting(userInput);

      // Switch between game modes based on user input
      switch (gameMode)
      {
      case EMode::CREATEMODE:
         CreateMode();
         break;
      case EMode::VIEWMODE:
         ViewMode();
         break;
      case EMode::QUIT:
         exit(0);
         break;
      case EMode::RETYPING:   // It returns without any function and receives the user's input again.
         break;
      }
   }
}
```

lib132.cpp (7217 bytes)

```
/*
* Minsu Jung
* CS 132 Project 1
* April 8
* A file that contains a function which, in turn, implements another function capable of interacting with user input.
*/
#include <iostream>
```

```cpp
#include <fstream>
#include <string>

// Enumeration for different game modes
enum EMode
{
   CREATEMODE = 1,      // Mode for creating a story
   VIEWMODE,         // Mode for viewing an existing story
   QUIT,          // Mode for quitting the program
   RETYPING       // Mode for retyping a story
};

// Function to display game start notification
void GameStartNotification();

// Function to check if a file exists
// This function takes a string argument representing a file name and returns if it exists
bool FileCheck(std::string fileName);

// Function to compare two strings while ignoring case sensitivity
// It takes in two string parameters, and
// returns a boolean value indicating whether the two strings are equal or not, ignoring the case of the characters.
bool EqualsIgnoringCase(std::string compareWord, std::string userInput);

// Function to set the game mode based on user input
// It takes a string values from user, the value refers the game mode.
// It returns the game mode as enum data type.
int GameModeSetting(std::string userInput);

// Function for creating a story
void CreateMode();

// Function for viewing an existing story
void ViewMode();

// Function to find and replace a placeholder in a string using user's input (from ReplaceString())
// It takes two string parameters to find Keyword using the tag( " < " ).
void FindingPlaceHolder(std::string fileContents, std::string outputFileName);

// Function to replace a placeholder in a string with user input
// It takes a string parameters to replace the keyword (in the file) to user's input
// , and returns user's input
std::string GetUserAnswer(std::string oldString);

// Function to write the story to a file
// It takes two string parameters "fileContents" has a whole string for a new story created by the user's answer.
// "outputFileName" is the name of ouput file that user made in a createMode();
void WriteStory(std::string fileContents, std::string outputFileName);


// This function checks if a file exists and returns a boolean value based on its availability.
bool FileCheck(std::string fileName)
{
   std::ifstream file(fileName);

   if (file.good())
   {
      file.close();
      return true;
   }
   else
   {
      std::cout << "File not found. Try again: ";
      return false;
   }
}

// This function sets the mode of the game and returns the appropriate value stored in ENUM according to the user input.
int GameModeSetting(std::string userInput)
{
   while (true)
   {
      if (EqualsIgnoringCase("create",userInput))
      {
         return EMode::CREATEMODE;
      }
      else if (EqualsIgnoringCase("view", userInput))
      {
         return EMode::VIEWMODE;
      }
      else if (EqualsIgnoringCase("quit", userInput))
      {
         return EMode::QUIT;
      }
      return EMode::RETYPING;
   }
}
```

```
// This function allows users to create a mad-lib story by reading from an input file
// and prompting them to fill in the placeholders.
void CreateMode()
{
   int startIndex = 0;     // index of " < "
   int endIndex = 0;    // index of " > "

   std::string inputFileName;
   std::string outputFileName;
   std::string fileContents;     // store the strings from the file
   std::string fileReader;       // Read the line from the file
   //std::string placeHolder;    // hold the word inside of " < >  "

   std::cout << "Input file name: ";
   std::getline(std::cin,inputFileName);

   while (!FileCheck(inputFileName))
   {
      std::getline(std::cin, inputFileName);
   }

   if (FileCheck(inputFileName))
   {
      std::cout << "Output file name: ";
      std::getline(std::cin,outputFileName);
      std::cout << std::endl;
      //kstd::cin.ignore();   // GetUserAnswer has getline() so it needs.
   }

   std::fstream InputFile(inputFileName); // Create input file


   while (std::getline(InputFile, fileReader)) // store the words in the file to fileContents
      fileContents += fileReader + 'Wn';

   FindingPlaceHolder(fileContents,outputFileName);

   InputFile.close();

}
// This function finds the placeholders in the input file
// and prompts the user to fill them in with their own words or phrases.
void FindingPlaceHolder(std::string fileContents, std::string outputFileName)
{
   std::string oldString;
   std::string replacedword;  // store replaced word from
   char startTag = '<';
   char endTag = '>';
   int startPos = 0;
   while ((startPos = fileContents.find(startTag, startPos))!= std::string::npos)
   {
      int endPos = fileContents.find(endTag, startPos);
      if (endPos == std::string::npos)
         break;

      oldString = fileContents.substr(startPos+1, endPos - startPos -1);
      replacedword = GetUserAnswer(oldString);
      fileContents.replace(startPos, endPos - startPos + 1, replacedword);
      startPos += replacedword.length();
   }
   std::cout << "Your mad-lib story has been created!" << std::endl;
   std::cout << std::endl;
   WriteStory(fileContents,outputFileName);
}

// This function prompts the user to input a word or phrase to replace a placeholder in the mad-lib story.
std::string GetUserAnswer(std::string oldString)
{
   std::string newString;
   std::cout << "Please type ";

   if (oldString[0] == 'a')   // if the first letter of a word is a,
   {
      std::cout << "an ";
   }
   else
   {
      std::cout << "a ";
   }

   std::cout << oldString << ": ";
   std::getline(std::cin, newString);
   return newString;
}

// This function writes the completed mad-lib story to an output file.
```

```
void WriteStory(std::string fileContents, std::string outputFileName)
{
   std::ofstream outFile(outputFileName);
   outFile<< fileContents;
   outFile.close();
}

// This function allows users to view a previously created mad-lib story according to user's inputfilename.
void ViewMode()
{
   std::string inputFileName;
   std::cout << "Input file name: ";
   std::getline(std::cin,inputFileName);

   while (!FileCheck(inputFileName))
   {
      std::getline(std::cin, inputFileName);
   }
   std::cout << std::endl;

   std::ifstream inputFile(inputFileName);
   std::string line;
   while (std::getline(inputFile, line))
   {
      std::cout << line << std::endl;
   }
   std::cout << std::endl;
   inputFile.close();
}

// This function checks if two strings are equal, ignoring their cases.
bool EqualsIgnoringCase(std::string compareWord, std::string userInput)
{
   if (compareWord.size() != userInput.size())return false;
   for (int i = 0; i < compareWord.size(); i++)
   {
      if (tolower(compareWord[i]) != tolower(userInput[i]))return false;
   }
   return true;
}

// This function displays a welcome message to the user at the beginning of the game.
void GameStartNotification()
{
   std::cout << "Welcome to the game of Mad Libs." << std::endl;
   std::cout << "I will ask you to provide several words" << std::endl;
   std::cout << "and phrases to fill in a mad lib story." << std::endl;
   std::cout << "The result will be written to an output file." << std::endl << std::endl;

}
```

mymadlib.txt (657 bytes)

```
Once upon a time, there was a <adjective> princess named <name>.
She lived in a <adjective> castle made of <noun> in the land of <place>.
One day, while wandering in the forest, she stumbled upon a group of <plural noun>.
They were dancing and singing around a <noun> and invited the princess to join them.
As she danced, she felt <adjective> and free, forgetting all her royal duties.
But when she returned to the castle, she was scolded by her <adjective> mother, the queen.
From that day on, the princess would sneak out to the forest to dance with the <plural noun>,
but always made sure to return to her <adjective> castle before anyone noticed.
```