CSE446/598 Assignment 9 (50 Points)
Spring 2020

Due: Saturday, April 25, 2020, by 11:59pm (Arizona Time), plus a one-day grace period.

Introduction

The aim of this assignment is to make sure that you have read the slides and the text and have understood the concepts covered in the lectures and in the text, including big data management and analytics, Hadoop, recommendation system, ontology, semantic Web, and cloud computing. By the end of the assignment, you should have a good understanding of the concepts and have applied these concepts in developing operational programs.

Practice Exercises (No submission required)

1. Reading: Textbook chapter 11, and lecture slides.

2. Answer the multiple choice questions in the Exercises and Projects part in chapter 11. Study the material covered in these questions can help you to prepare for the class exercises, quizzes, and the exam.

3. Study for the short answer questions in the Exercises and Projects part in chapter 11. Make sure that you understand these questions and can briefly answer these questions. Study the material covered in these questions can help you to prepare for the exam and understand the homework assignment.

4. Study Textbook section 11.3 on Big Data Processing and 14.8 on Hadoop. Make sure you understand how the program works. Implement the program on the Hadoop environment that you have installed.

5. Use VIPLE to calculate the average of 1, 2, 3, …, 2000 using a single thread and using four parallel threads.

6. Hadoop Exercise.
   a. Install Hadoop on a Linux computer.
   b. Modify the WordCount program given in the lecture slides or Textbook section 11.2. You must add time stamps at the beginning and the end of the Map part of the program, and at the beginning and the end of the Reduce part of the program, so that you can measure the time consumed in each part of computation. You must add print statements to print the four time stamps and the time consumed in each part as your program output. The program must also print the number of words counted.
   c. Create a text file as the input file to test the WordCount program in your Hadoop environment.
   d. Instruction for testing your program.

Assignment Questions (Submission Required, 50 points)

There are two questions in this assignment. You choose one question of the two questions to implement. If you implement both questions, we will grade question 1 only.

1. In this assignment, you will create a Web application platform that allows user to perform automated parallel computing that mimics the Hadoop process. A sample user interface is given in Figure 1.

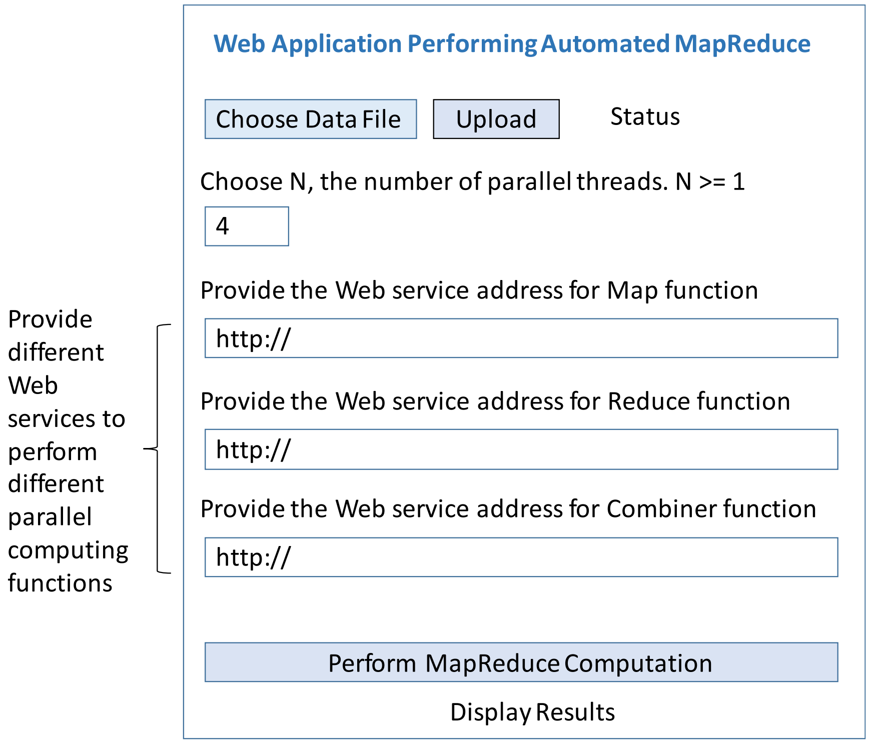

Web Application Performing Automated MapReduce
Choose Data File
Upload
Status
Choose N, the number of parallel threads. N >= 1
4
Provide the Web service address for Map function
http://
Provide the Web service address for Reduce function
http://
Provide the Web service address for Combiner function
http://
Provide different Web services to perform different parallel computing functions
Perform MapReduce Computation
Display Results

Figure 1. Web application GUI with Web service calls

The Web application consists of the following components.

- GUI component and the connection to the other components. [10 points]
- NameNode: It splits the uploaded Data File into N subsets and provides a subset to each TaskTracker. [10 points]
- TaskTracker: It consists of Map function and a Reduce function.
- Map function: It converts data from subset of Data File into Key-value pairs, based on the functionality requirement. [10 points]
- Reduce function: It reduces the key-value pairs into one or smaller number of key-value pairs, based on the functionality requirement. [10 points]
- Combiner: It synchronizes and combines the results from all the Reduce functions and put the final results. [10 points]

In this assignment, you will implement the Word Count application. It counts the number of occurrences each word in the uploaded Data File. The components and the computation process is illustrated in Figure 2.

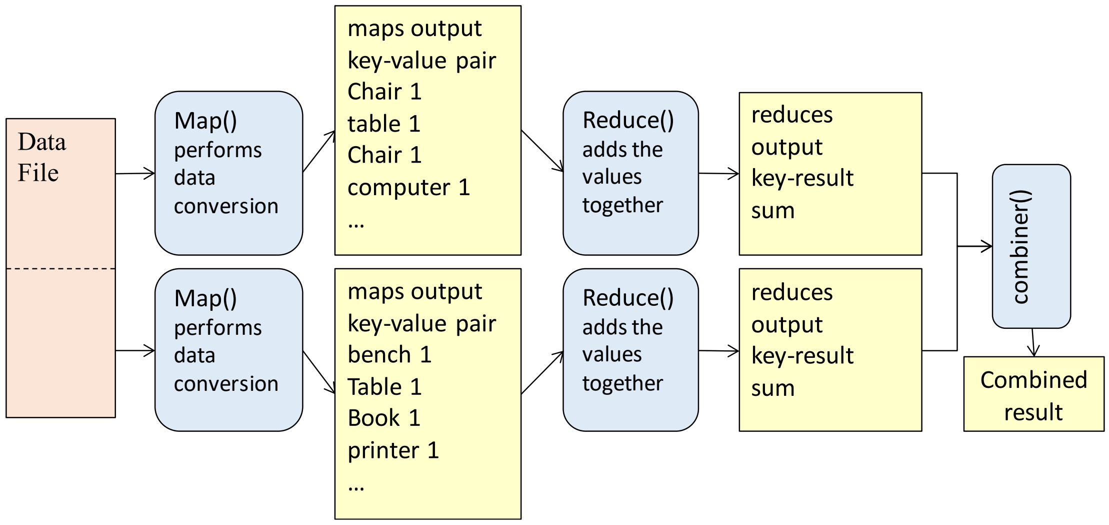

Data File
Map() performs data conversion
Map() performs data conversion
maps output key-value pair
Chair 1
table 1
Chair 1
computer 1
...
maps output key-value pair
bench 1
Table 1
Book 1
printer 1
...
Reduce() adds the values together
Reduce() adds the values together
reduces output key-result sum
reduces output key-result sum
combiner()
Combined result

Figure 2. Map and Reduce execution with N = 2

You can implement the Web application either as an ASP .Net Website application, HTML5 application, or MVC Web application on localhost. For the components used in the application, you can also choose to implement this application in one of the following options.

Option 1: You implement the Web application as a service-oriented application. You implement Map, Reduce, and Combiner as Web services (either RESTful services or WSDL services). You make parallel calls to the services for each Map and Reduce, and you configure or assume that a new instance of service will be created for each call, and thus the tasks are done in parallel. The sample GUI given in Figure 1 is for Option 1.

Option 2: You implement the Web application without calling Web services. You implement Map, Reduce, and Combiner as local components. You must use multithreading techniques (read Text Chapter 2) to run the Map and Reduce functions in parallel. In this option, a large Data file, with at least 5000 words must be used. The execution times for single thread execution and for multithread execution must be reported. A sample GUI for Option 2 is given Figure 3.

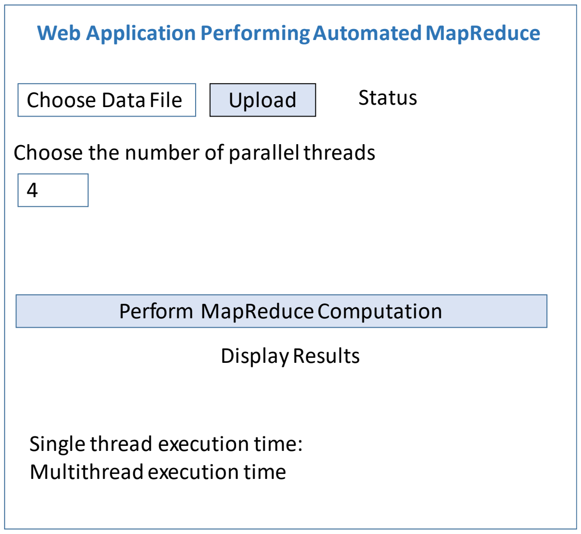

Web Application Performing Automated MapReduce
Choose Data File
Upload
Status
Choose the number of parallel threads
4
Perform MapReduce Computation
Display Results
Single thread execution time:
Multithread execution time

Figure 3. Web application GUI without Web service calls

2. In this question, you will implement the same function as described in question 1. However, you will use VIPLE, to simulate the Hadoop process, instead of using a web application. Figure 4 and Figure 5 give a sample application, where string's characters are counted. It is not a realistic application that improves the performance of computing. The main purpose of this example is to show how you can automatically start N threads to perform parallel computing, where N can be entered from the keyboard. In your assignment question, you will count words, just like described in question 1, instead of counting characters.

   The key idea of generating N threads based on the input N is to use event-driven programming. The Custom Activity NameNode will generate an event output (circular output port in Figure 5), instead of a data output (triangular port in Figure 5). Once the event output port is used, it will trigger an event through the Custom Event activity in the Main Diagram every time it is executed. As can be seen in Figure 5, there is no computation inside the activity. Its purpose is to trigger a new thread.

   In the Main diagram, the NameNode is placed in a loop, which will be called N times. Each time will trigger a new thread through the Custom Event activity. The main computations are performed in these parallel threads. The loop will be executed sequentially. Since we do not have much computation in the activity, it will be iterated and the parallel threads are generated without delay.

   You tasks in the option are as follows. You can make use of the given code as your starting point.

   - Write a service, a code activity, or a custom activity to load the text in a text file into variable InputData. Use it to replace the Data activity that provided the Input data. [10 points]
   - Write a service, a code activity, or a custom activity to counts number of appearances for each word in the InputData. [10 points]
   - In the main diagram, you must automatically create the threads based on the input N, as shown in Figures 4 and 5. [10 points]
   - Write a code activity to save the result (the number of words) into a text file called FileForOutput.txt. [10 points]

- Make sure that your program can execute correctly even if the data length is not a multiple of N. In the given example program in Figure 4, this check is not performed to keep the program small. [10 points]

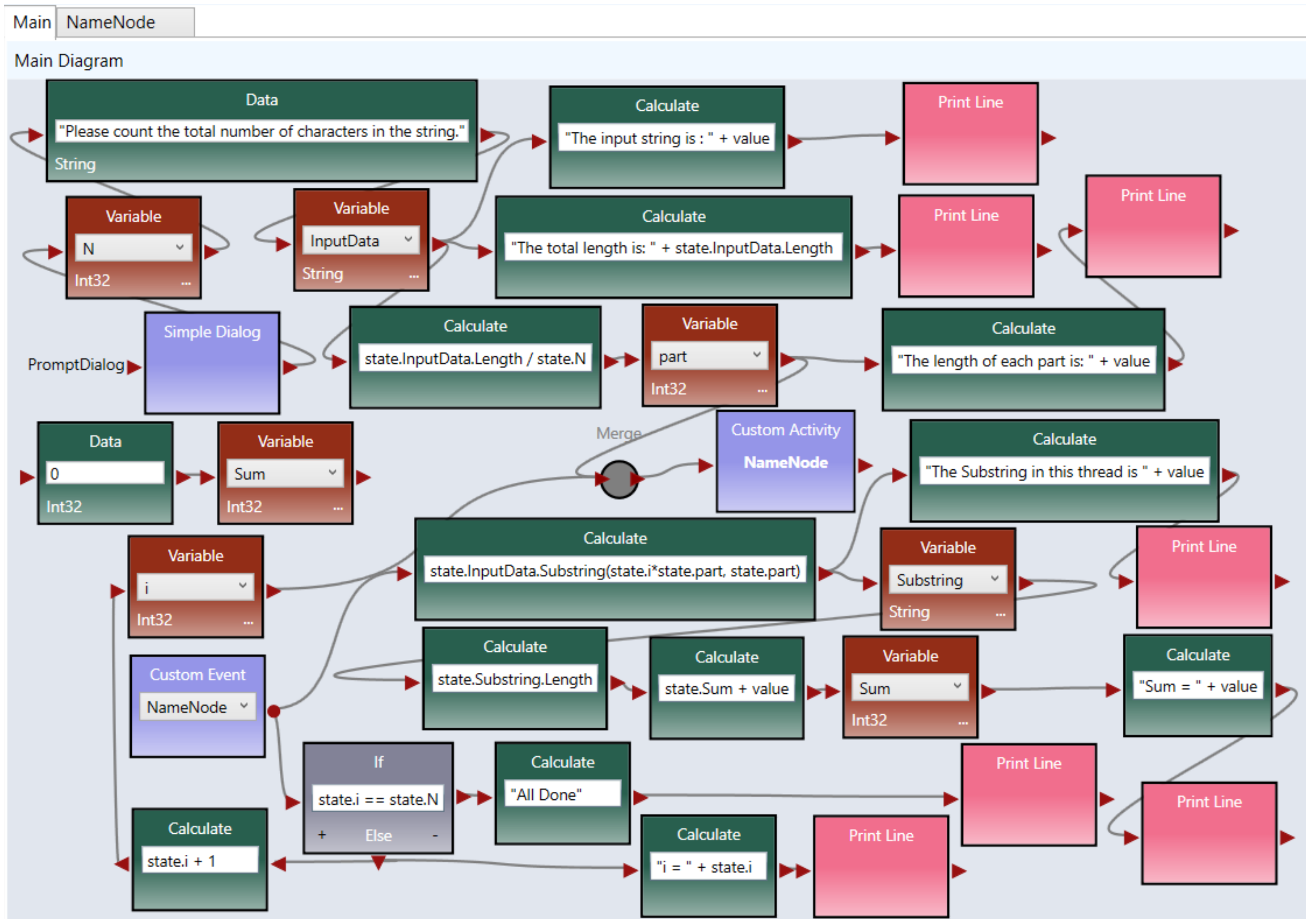


Figure 4. The Main diagram of the automated parallel threads example

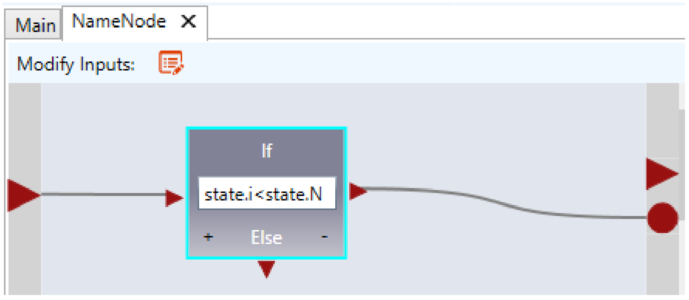
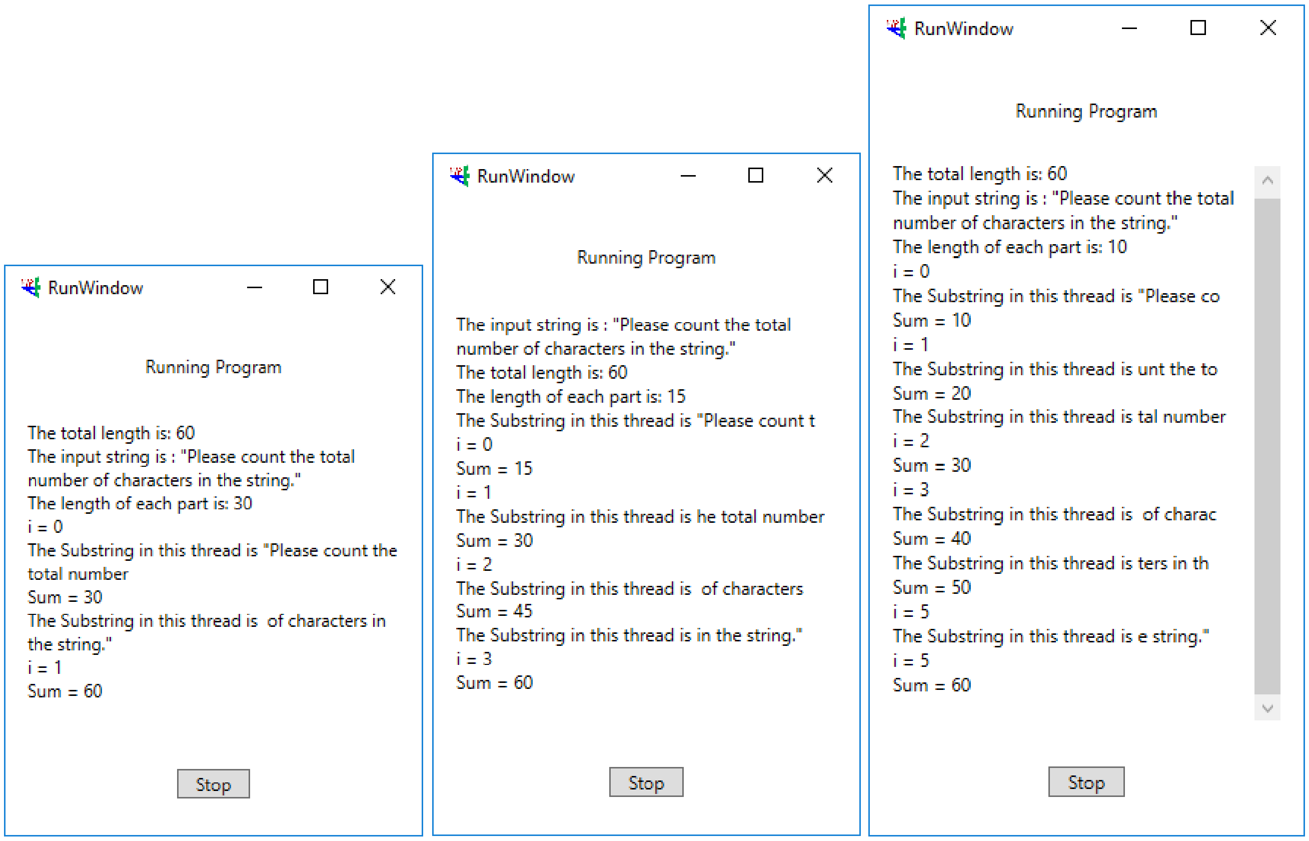


Figure 5. The Custom Activity diagram of the automated parallel threads example

Figure 6 shows the execution output when the input number N is 2, 4, and 6, respectively.

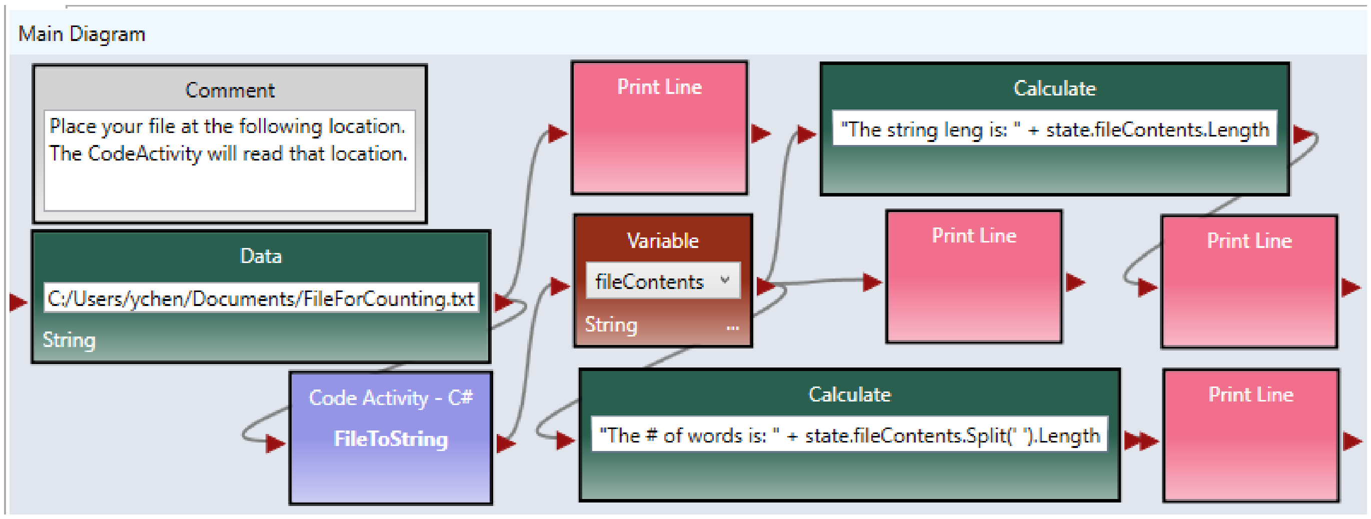


```
The total length is: 60
The input string is : "Please count the total
number of characters in the string."
The length of each part is: 30
i = 0
The Substring in this thread is "Please count the
total number
Sum = 30
The Substring in this thread is  of characters in
the string."
i = 1
Sum = 60
```

```
The input string is : "Please count the total
number of characters in the string."
The total length is: 60
The length of each part is: 15
The Substring in this thread is "Please count t
i = 0
Sum = 15
i = 1
The Substring in this thread is he total number
Sum = 30
i = 2
The Substring in this thread is  of characters
Sum = 45
The Substring in this thread is in the string."
i = 3
Sum = 60
```

```
The total length is: 60
The input string is : "Please count the total
number of characters in the string."
The length of each part is: 10
i = 0
The Substring in this thread is "Please co
Sum = 10
i = 1
The Substring in this thread is unt the to
Sum = 20
The Substring in this thread is tal number
i = 2
Sum = 30
i = 3
The Substring in this thread is  of charac
Sum = 40
The Substring in this thread is ters in th
Sum = 50
i = 5
The Substring in this thread is e string."
i = 5
Sum = 60
```

Figure 6. The output when the input number N is 2, 4, and 6, respectively.



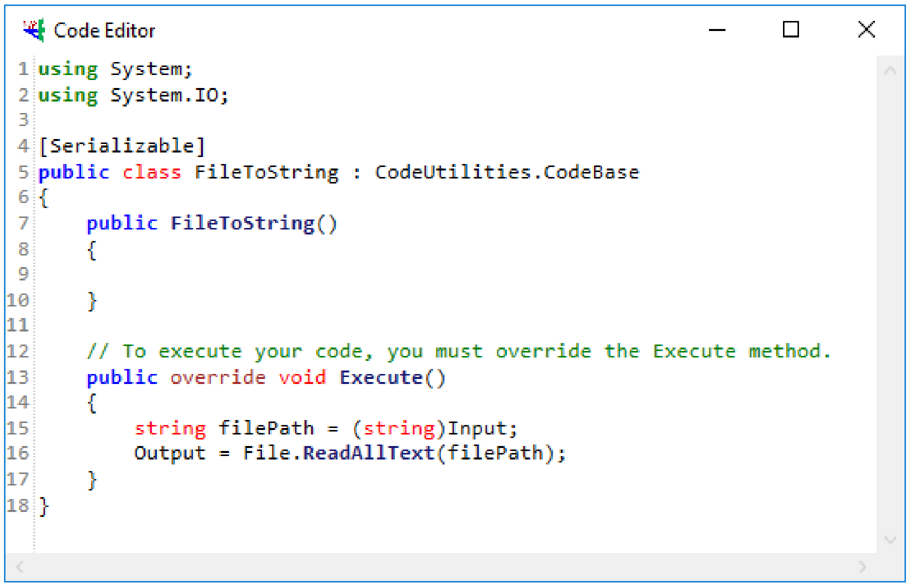
Place your file at the following location.
The CodeActivity will read that location.

C:/Users/ychen/Documents/FileForCounting.txt

"The string leng is: " + state.fileContents.Length

"The # of words is: " + state.fileContents.Split(' ').Length

Figure 7. Sample application that reads a text file into a string.



```
using System;
using System.IO;

[Serializable]
public class FileToString : CodeUtilities.CodeBase
{
    public FileToString()
    {

    }

    // To execute your code, you must override the Execute method.
    public override void Execute()
    {
        string filePath = (string)Input;
        Output = File.ReadAllText(filePath);
    }
}
```

Figure 8. The CodeActivity reading a text file into a string.

General Submission Requirement

All submissions must be zipped into a single file electronically submitted to the Canvas assignment folder. All files must be zipped into a single file for submission.

For programming assignments, the entire solution folder with all the files must be included. To make sure that you have all the files in the zip file, unzip the file on a different machine or in a different directory, and test if you can run the project in a different location.

Submission notice: A programming assignment typically consists of multiple distributed parts. They may be stored in different locations when you create them. You must copy these projects into a single folder for Canvas submission. To make sure that you have all the files included in the zip file and they work together, download your own submission from the Canvas. Unzip the file on a different machine, and test it and see if you can run the solution in a different location because the TA will test your application on a different machine.

Grading and Rubrics

Each sub-question (programming tasks) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each sub-question, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% - 10%	0
Each sub-question	Meeting all requirements, well commented, and working correctly in all test cases	Working correctly in all test cases. Comments not provided to explain what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed compilation, showing some effort, but the code does not implement the required work.	No attempt

Late submission deduction policy:

- Grace period (Sunday): No penalty for late submissions that are received within 24 hours of the given due date;
- 1% grade deduction for every hour after the first 24 hours of the grace period!
- No submission will be allowed after Tuesday midnight. The submission link will be disabled.