CSE445/598 Project 3 (Assignments 3&4) (50+50 Points)
Fall 2023

Assignment 3 Due by 11:59pm of September 30, 2023, in Canvas only.
Individual and Teamwork Submissions.

Assignment 4 Due by 11:59pm of October 14, 2023, in WebStrar Server and in Canvas.
Individual and Teamwork Submissions.

One day grace period for each assignment applies.

Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures, including service development, service registration, service deployment, service hosting, service proxy, service binding, service invocation, and application building using your own services and public services. By the end of the assignment, you should have applied these concepts in programming your services, deploying your services, and have used your own services and public services to compose your SOC applications.

This project is partly an **individual project** (about 75%) and partly a **group project** (about 25%). Team formation is required! Each project team will consist of two or three members, based on the team building exercise. Although a team works together to complete the project in a collaborative and coordinated manner, a large part of the project will be done and graded individually. A declaration must be given, which specifies the portion of individual efforts in the joint part of the project. A percentage of contribution of each member (e.g., 35%, 35%, and 30%) must be given for the joint part of work, which will be used to **scale** the assignment grades of the group part. Each service must have a single author associated with it. The final project must also be deployed into the given Web server: WebStrar. If you choose to develop the services in Java, you must use your own server, for example, using an AWS free tier server. In this case, you must create and configure your server.

When submitting the services and service test pages (TryIt pages) into the Canvas, you must submit the folder with the source code (We will read the source code from Canvas and test the code from the server). For the server deployment, you can deploy your project with the source code (Just-In-time Compilation) or with the precompiled files in the folder. Please carefully read WebStrar Tutorial before deploying your project into the server.

Practice Exercises (No submission required)

No submission is required for this part of exercises. However, doing these exercises can help you better understand the concepts and thus help you in quizzes, exams, as well as the assignment questions.

1. Reading: Textbook Chapter 3 and Appendix C.

2. Answer the multiple choice questions 1.1 through 1.16 of the text section 3.10. Study the material covered in these questions can help you prepare for the class exercises, quizzes, and the exams.

Answer keys to the questions can be found in the course web page. To better learn these concepts, you should do the exercises based on your understanding, and then check the answer keys.

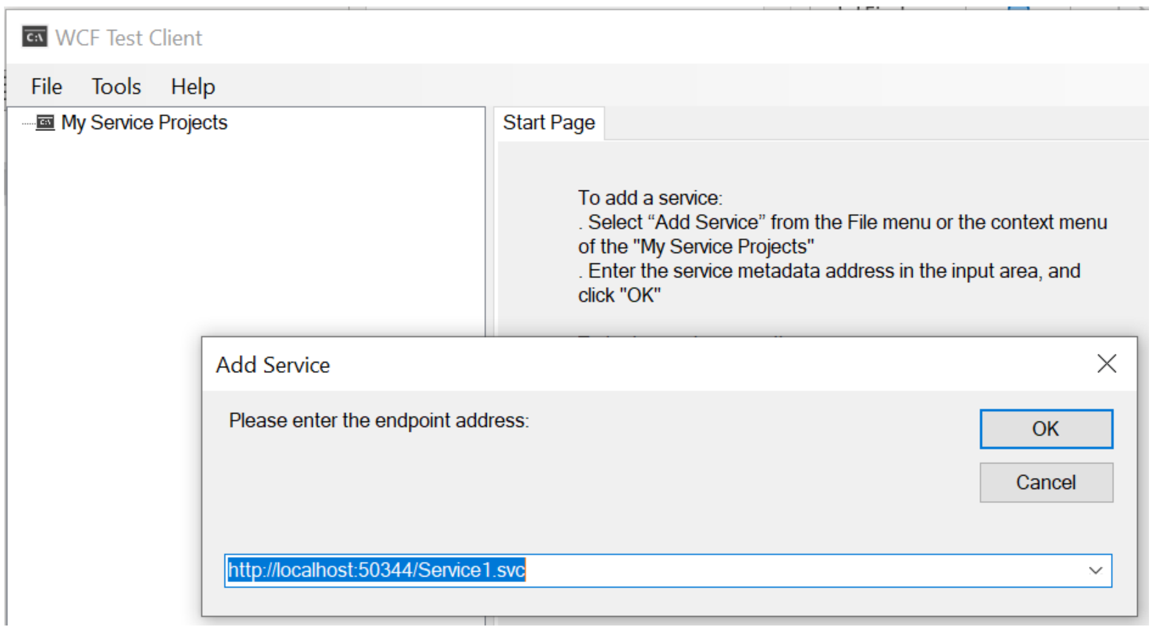
3. Study for questions 2 through 16 in text section 3.10. Make sure that you understand these questions and can briefly answer these questions. Studying the material covered in these questions can help you prepare for tests and understand the homework assignment.

4. Questions 17 through 19 are largely covered by the assignment questions in Part II.

5. Web Service Testing. You can test a WSDL service before you add it to your Web application, whether the service is found in a public repository or is developed by yourself. When you start a WCF service in Visual Studio, WcfTestClient should start automatically. You can also manually start it. It is located in your Visual Studio folder:

```
C:\Program Files(x86)\Microsoft Visual Studio\2022\Community\Common7\IDE/WcfTestClient.exe
```

Use the tool to test the services that you developed or discovered in public repository. To test a service, use its menu: File → Add Service:



To add a service:
. Select “Add Service” from the File menu or the context menu of the "My Service Projects"
. Enter the service metadata address in the input area, and click "OK"

Please enter the endpoint address:

http://localhost:50344/Service1.svc

Project/Assignment Questions (Submission required, 50+50 points total)

The purpose of this project is to exercise service development, service discovery, remote binding, application composition using your own services and external public services. You will also exercise service and application deployment in this project. Some of the services to be developed can be synthetic, e.g., banking service, while some should be realistic, e.g., en/decoding, en/decryption, and product catalog. However, you should make your overall application as realistic as possible. The project will be completed in two assignments, each with multiple parts.

Project 3 Assignment 3

This assignment includes Questions 1 (group) and 2 (individual)

Q1 For group. **Requirement Document**. The requirement document will be prepared by all team members together and the same (identical) document must be included in each and every team member's assignment 3 submission. The document must contain the following contents. **[10 points]**

1.1 Description of the service-oriented computing system that your team plans to develop. Simply taking an idea that you found from the Internet or from others is not acceptable. Your team must have some of your own ideas. [2]

1.2 A diagram showing the overall system design, its layers, components, and the connections among the services. A sample diagram is given in Figure 1. You must come up with your own system and diagram. You will not implement the application logic layer of the system outlined here in your document. It will be implemented in Project 5 later. The focus of Project 3 is to develop and deploy the **services** outlined in this requirement document. Thus, you must pay more attention to the services in this project. [2]

1.3 Create a **service directory** (a table) listing the services that your team plans to develop, including the required and elective services, including at least 2 required services. Each service must be described in such detail that shows the feasibility of implementing the service in a reasonable amount of time, e.g., 10 hours. All the elective services must be directly related to the application that you outlined in the previous question and support the composition of the application. The required services do not have to be related to your application, but you should try to develop those services that could be used in your team's application. The contents of the service directory below are examples, and you need to fill out the table based on your own project plan. [6]

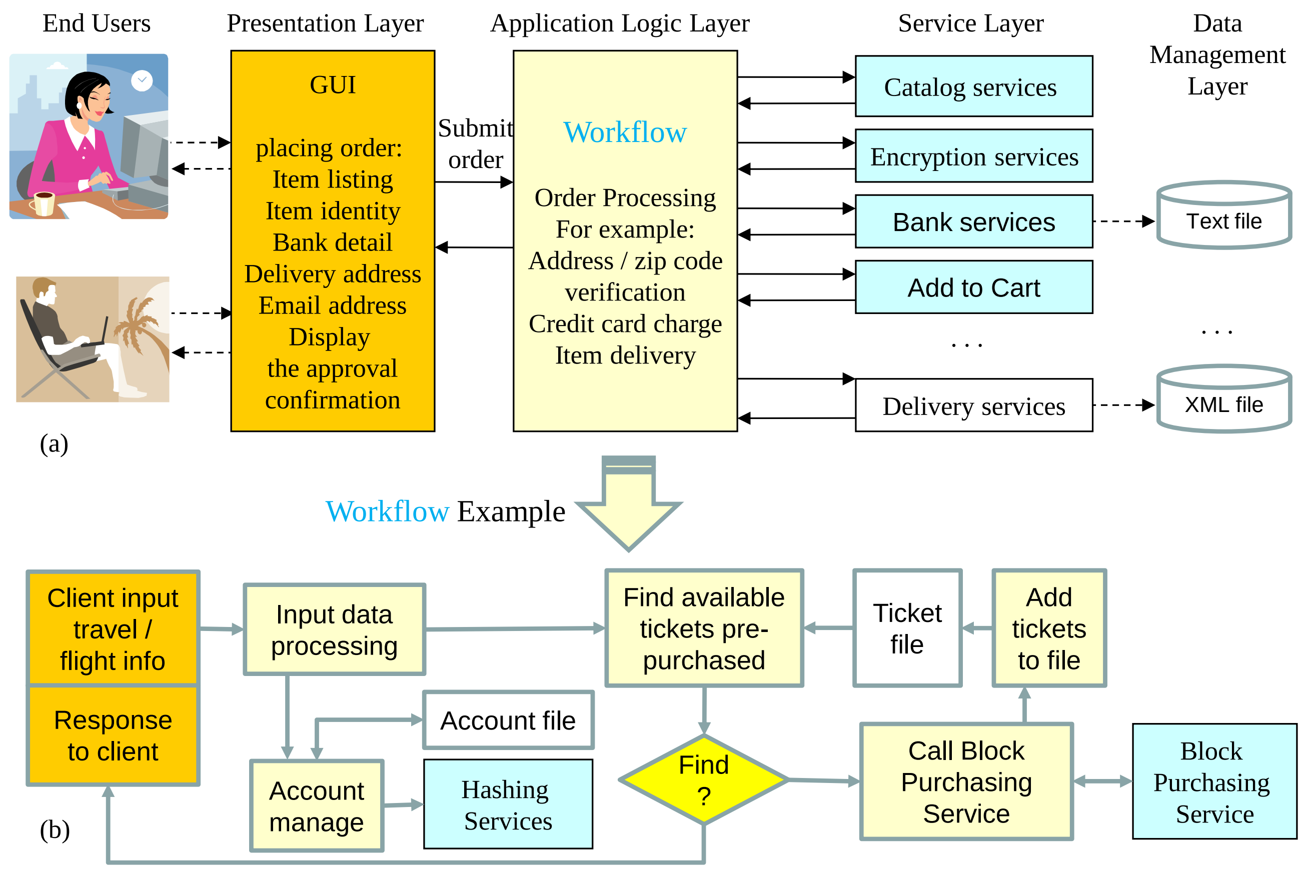


Figure 1. (a) A sample diagram of a four-tier service-oriented computing system and (b) a sample workflow of an example system

You must replace this diagrams using the diagram that you plan to develop.

	Service Directory: The team plan to complete the following services. Changes can be made later in the final project (Project 5).					
	This service directory will be deployed at this address*:					
	Team name/Number: Put your team name here.					
Provider name	Service name, with input and output types	Required service?	Difficult level – self estimate	TryIt link*	Service description	Planned resources need to implement the service
Member 1 name	Encryption and decryption: Input: String Output: String	Yes or No	Easy, moderate, or chall-enging	TryIt link*	Cipher encryption and decryption of each character of the input string	Write my own code and use local component to implement the service
Member 1 name	SolarPower Inputs: zip code and size Output: integer	Yes or No	moderate, or chall-enging	TryIt link*	Output annual KW number for a given solar panel size at a given zip code location	Retrieve information from national database at: http://graphical.weather.gov/xml/
Member 1 name	findStore Input: zipcode Output: list of string	Yes or No	moderate, or chall-enging	TryIt link*	Use an existing online service or API to find the locations of a given store name	Use the service from Yelp site at: http://www.yelp.com/
…	…			…	…	…
Member 2 name	textToPhone inputs: string Account, string Password, string Receiver, string Subject, string Body Output: boolean	Yes or No	moderate, or chall-enging	TryIt link*	Send a text to a cell phone and return if the message is sent successfully	Use google gmail account and carrier services: phone_number@tmomail.net phone_number@messaging.sprintpcs.com phone_number@vtext.com phone_number@txt.att.net
…	…			…	…	…
Member 3 name	Service name …	Yes or No	moderate, or chall-enging	TryIt link*	Service does this work	Code here
…	…			…	…	…

*Not required in assignment 3 submission, but is required for A4 submission.

Assignment 3 Question 2: This question is an individual assignment.

Q2 Required Services and (TryIt) Test Pages [40 points]

In this assignment, you must implement a set of services. There are two types of services to be developed: required services and elective services. The required services will be developed in assignment 3 (this part) and the elective services will be developed in A4.

2.1 Two Required Services [30 = 15+15]

A set of required services and their requirements are listed in a separate document named "List of Required Services". Each team member must choose and implement **two** services from the given list. The members in the same team must choose **different services** from the given list. Slight variation (modification) to the service is allowed, as long as the variation does not significantly simplify the task of developing the service. You can implement the services either as WSDL services or as RESTful services.

2.2 TryIt (Test) Page of the Required Services [10]

What is a TryIt page? It is a simple Web application that allows you to test your services by entering inputs and displaying outputs. The TryIt page mainly provides a GUI for the input and output of the Web services or APIs that you have developed. The difference between a TryIt page and a normal Web application is that a TryIt page focuses on showing the functionality of individual services (like unit testing in software testing), while a Web application focuses on providing the overall functionality of the application that makes use of the services and components (like integrated testing in software testing).

For each service and operation that you developed, you must include its test GUI in the **TryIt** page to allow the human user to test the service. The TryIt test page must contain the following contents for each service:

(A) A sentence to describe the functions of each service (operation)

(B) The URL of the service

1. For assignment 3 submission: use the localhost URL of the service.
2. For the final submission in A4, the WebStrar server URL of each service must be used in the TryIt page.

(C) Method (Operation) names, with parameter type list and the return type for each endpoint.

(D) Text boxes for entering inputs

(E) Invoke buttons to call the services

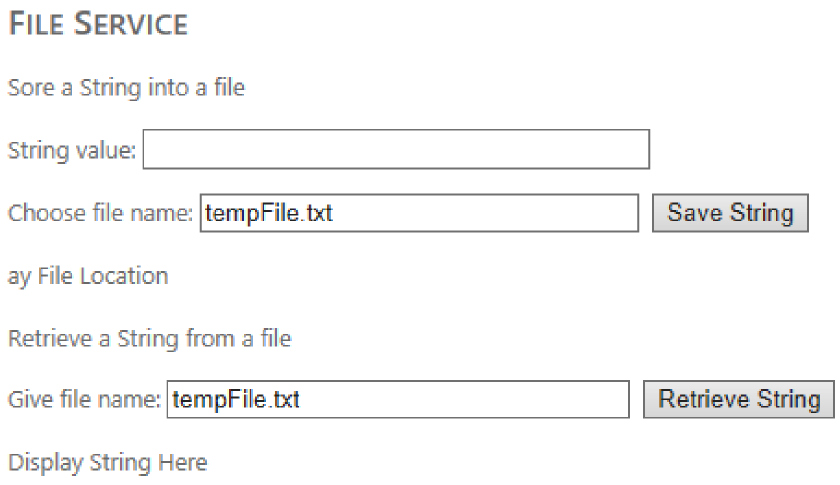
(F) A place (e.g., label or listbox) to display the service response (output)

You can also develop a different TryIt page for a different service. You can develop a combined TryIt page for all services. In this project, you are required to develop a combined TryIt for all services. The reason for this requirement is you will not have sufficient folders to host multiple TryIt pages in WebStrar.

The services and their TryIt page(s) will be tested on localhost in the assignment 3 submission. To make sure the TA can test your services and TryIt page on their computers, you must download your submission and test your submission on a different computer before submission. An easy mistake in submission is not to include all the required files in the submission, and thus the TA cannot run your project.

You must combine all the TryIt pages of multiple services and operations into a single TryIt page. You must make sure that you have a GUI part to test every service operation that you developed for credit. An example of a TryIt page is given in the following figure, which us available at the following link:

http://venus.sod.asu.edu/wsrepository/Services/FileServiceTryIt/



More examples of TryIt test pages are at:

http://venus.sod.asu.edu/WSRepository/CoffeeMachine/

http://venus.sod.asu.edu/WSRepository/Services/ImageVerifierSvc/TryIt.aspx

http://venus.sod.asu.edu/WSRepository/RandomStringTryIt/TryIt.aspx

http://venus.sod.asu.edu/WSRepository/services/wsTesterTryIt/

Assignment 3 Questions 1 (1.1, 1.2, and 1.3) and Question 2 (2.1 and 2.2) Submission into Canvas:

Individual submissions: **Each** team member must submit both Questions 1 and 2. The submission must include:

(1) A PDF file that answers Question 1 (1.1, 1.2, and 1.3) (All members submit the same file), and
(2) Your solutions/Projects for Question 2 (Each member submits your services and TryIt solutions/Projects). These files must contain all the **required services** and the associated TryIt page developed by that member in this assignment. Do not include other member's services and TryIt page! All the solutions must be included in a single zip file and submit the zip file into **Canvas**. The solutions must be testable (executable) on a different computer (TA's computer).

Project 3 Assignment 4

This assignment has a different due date.

Assignment 4: Elective Services Development: Individual assignment and teamwork [50 points]

This assignment consists of two questions, each member will develop different elective services and their TryIt page.

Q3. Elective services and TryIt page

3.1 **Elective Services**: [25 points for 3.1]

Each member must develop at least two elective services (or service operations). For the elective services, the team members must discuss what services are needed based on the application defined in Assignment 3, or a revised version of the project (you can make changes to the service list submitted in in assignment 3), and who should develop what services. All team members must define and implement different elective services. You can use additional required services as your elective services if the service is related to your application. For each team member, least one service must be a WSDL service and at least one service must be a RESTful service. For the rest of the services, you can choose to develop WSDL services or RESTful services. For developing RESTful services, please follow Lecture Slides L10 or textbook section 7.3.3.2 for more detail.

The difficulty level of the elective services (operations) that you developed will be rated by the instructor and the teaching assistant into one of the following three difficulty levels:

(1) **Easy**: The method (operation) in this service implements a simple math function and can be done using less than 50 lines of code, such as a Fahrenheit and Celsius temperature conversion. [5 points each]

(2) Moderate: There are algorithmic issues to address and the code for each method will be at least 50 lines, such as self-developed encryption/decryption, efficient sorting, or equation system solving. If a service operation can be done in less than 50 lines, but you use more than 50 lines, it will be counted as an easy service. [7.5 points each]

(3) Challenging: Services that require at least 50 lines of code and use states or files, such as creating a simulated (synthetic) banking service that allows users to sign up, create an account, deposit funds, spend funds, etc.; or services that make use of multiple available services (operations) or APIs provided by other providers, such as Amazon's services, Microsoft services (e.g., Bing map service), Google code's APIs, or the ASU services. These services and APIs may or may not have WSDL interfaces. Your services must provide WSDL or RESTful interfaces. The data received from other services should be processed and combined before returning it to the clients. The given required services can be used as the basis of your challenging services. You can add functionalities to make it meet the challenging requirement. Database is not allowed in this assignment. If you need to store states, you can use either a text file (See Chapter 3 or L12 Slides) or an XML file (Read Text Chapter 4 and also Chapter 5, Section 5.4). [12.5 each]

To obtain the full points in question 3.1, you need to develop at least two elective services and at most four elective services. It implies that you cannot write five easy services in this question. If you write easy services only, the maximum number of points you can obtain in this question is 20 points. On the other hand, you can obtain 25 points at most, even if you develop more services and more difficult services than required. From the format point of view, you can either define these services methods in one big service, or define them as separate services. Each service and method must be commented in detail, including the functionality, parameters, types, and the return value type. Again, due to WebStrar limitation, you may need to combine multiple services into one service consisting of multiple service operations.

3.2 Development of TryIt Page for Elective Services [5 points for 3.2]

You must add the test GUI items to the elective services into your TryIt page: One TryIt page for required services and elective services. The requirement is the same as defined in question 2.2 in assignment 3, but all the URLs must be based on the WebStrar server URLs because the services have been deployed to the server. You must also have the URL of the Main page (See next question) linked to the TryIt page, so that one can return to the main page from the TryIt page.

3.3 Deployment of Required Services, Elective Services, and TryIt page [5 points for 1.3]

All the services (required and elective) and their TryIt page must be deployed into the WebStrar server. You must deploy the services first. Before deploying the TryIt page, you must change all the service references from localhost addresses to the server addresses, so that the services can be tested from the server.

Q4 Main Page and Server Deployment (group assignment question) [15 points]

The final **service directory** must be an html page, which must be named index.html, so that the page can be accessed through the link: http://webstrarX.fulton.asu.edu/index.html, where X is your WebStrar site number. This means that the index page will be deployed in the root directory of your WebSrar server. You **must** include this link into your Canvas submission so that we can find your server entry address. The **service directory** (main page) must contain the following contents:

4.1 List all **required** and **elective** services from all members: links to the test pages (TryIt page) and services. The schema and a list of example services are shown in the table below, which is an updated version of the table created in assignment 3. In this table, the TryIt pages must be linked into the directory, and all the links must be the server links. No local host links can be used. [8]

	Service Directory: The team has completed the following services.					
	This page is at: http://webstrarX.fulton.asu.edu/index.html (change X to your WebStrar site #)					
	This project is developed by: Put your team member's names here.					
Provider name	Service name, with input and output types	Required service?	Difficult level – self estimate	TryIt link	Service description	Actual resources used to implement the service
Provider name: ASU repository	Encryption and decryption: Input: String Output: String	Yes or No	Easy, moderate, or chall-enging	TryIt	Cipher encryption and decryption	Use library class and local component to implement the service
Member 1 name	SolarPower Inputs: zip code and size Output: integer	Yes or No	moderate, or chall-enging	TryIt	Output annual KW number for a given panel size at a given zip code location	Retrieve information from national database at: http://graphical.weather.gov/xml/
Member 1 name	findStore Input: zipcode Output: list of string	Yes or No	moderate, or chall-enging	TryIt	Use an existing online service or API to find the locations of a given store name	Use the service from Yelp site at: http://www.yelp.com/
	…			…	…	…
Member 2 name	textToPhone inputs: string Account, string Password, string Receiver, string Subject, string Body Output: boolean	Yes or No	moderate, or chall-enging	TryIt	Send a text to a cell phone and return if the message is sent successfully	Use google gmail account and carrier services: phone_number@tmomail.net phone_number@messaging.sprintpcs.com phone_number@vtext.com phone_number@txt.att.net
…	…			…	…	…
		Yes or No	moderate, or chall-enging			

Make sure that you have the deployment URL in the first row of the page!

4.2 Deploy the table in html into the WebStrar server*, in the root directory outside the pre-created directories, so that the page can be accessed using the address: [7 points]

http://webstrarX.fulton.asu.edu/index.html (change X to you WebStrar site number)

This page and its URL in WebStrar must also be included in your Canvas submission by each member, among other files.

Deployment suggestion: Your team divides the pre-created pages (page0, page1, …, page10 in WebStrar) among the team members. One pre-created page can host one application or one service only. For each team member, you must deploy your TryIt application into a separate page. Then, deploy one service into one pre-created page. If you do not have enough pages to deploy all your services, you must combine multiple services into one service: The service contains multiple service operation – each operation is counted as a service.

Assignment 4 Submission

A4 Individual Submission into Canvas:

(1) **Each** team member must submit one zip file into **Canvas**. The zip file must contain the Solutions/Projects of your **elective services** and their TryIt page(s) developed in assignment 4. Do not include other team member's elective services and TryIt pages. The grader will use this submission to run your code locally and read your code of services and TryIt pages.

(2) The html page of your final **service directory**, which must is named index.html. The page should be open in Web browser and the links in the html page should point to your WebStrar deployment. The grader can use this html page to test your deployment.

A4 Team submission into WebStrar Server:

(1) All the services (required and elective from both assignments 3 and 4) and their TryIt pages must be deployed into the WebStrar server by each member. The required services (that have been submitted in assignment 3) do not need to be submitted into the Canvas, but must be deployed in WebStrar.
If you developed other types of services in Java or Python, you must deplay your services into cloud server of your own.

(2) The **Service Directory** table (index.html file) must be deployed into the WebStrar at the address: http://webstrarX.fulton.asu.edu/index.html (change X to you WebStrar site number). The index.html file should be same for individual submission and team submission.

Group assignment questions grading

Questions 1 in Assignment 3 and Question 4 in A4 are group questions. For each group question, the following grading scale will be applied:

The member grade *Gi* for member *i* is calculated and scaled through the formula:

$$G_i = S * \frac{N * P_i}{\sum_{i=1}^{N} P_i}$$

where

G_i: The grade of member i (G_i <= T. If G_i > S, set to T), where S is the points of the questions.

S: The score of the assignment question before scaling. For question 1, S = 10, For question 4, S = 15.

P_i: Contribution percentage of member i (P_i is between 1% and 100%). The team members should provide the percentages.

N: The number of active members in the team (N = 2 or 3). If a member dropped or does not make any contribution, the member does not count in calculation.

General Submission Requirement

All submissions must be electronically submitted to the assignment folder where you downloaded the assignment paper. All files must be zipped into a single file.

Submission preparation notice: The assignment consists of multiple **distributed** projects and components. They may be stored in different locations on your computer when you create them. You must choose your own location to store the project when you create the project. Then, you can copy these projects into a single folder for the blackboard submission. To make sure that you have all the files included in the zip file and they work together, you must **test** them before submission. You must also download your own submission from the blackboard. Unzip the file on a different machine and test your assignment and see if you can run the solution in a different machine, because the TA will test your application on a different machine.

If you submitted an empty project folder, or an incomplete project folder, we cannot grade your resubmission after the due date! We grade only what you submitted before the submission due date. Please read FAQ document in the course Web page for more details.

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 or 1 point
For 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 to compile, showing some effort, but the code does not implement the required work.	0 points if no submission 1 point if submitted files that do not answer any questions required, for example, the files are empty, cannot be open, wrong files, etc.

Late submission deduction policy:

- Grace period: 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 at 11:59pm on Tuesday. You must make sure that you complete the submission before 11:59pm. If your file is big, it may take more than an hour to complete the submission!