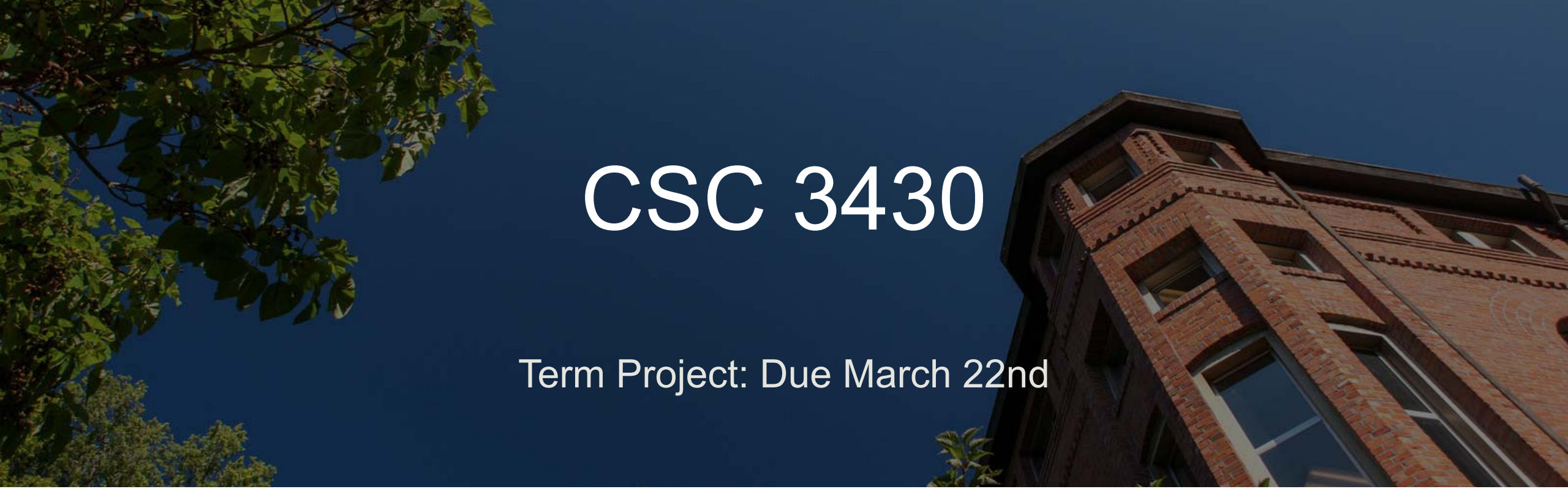

CSC 3430

Term Project: Due March 22nd

SEATTLE PACIFIC UNIVERSITY

NATIONALLY RANKED

SPU is the only private university in the Pacific Northwest to make *U.S. News & World Report's* 2018 "Best National Universities" list.

General Characteristics

- **It is a Programming Project**
- **You are allowed to use a third party library**
- **You may choose your programming language between Java and Python**
- **Extra credit if you animate the graph drawing as the algorithm process the members**
- **You will need to write a report**
- **It is a team activity (2 people per team)**
- **There will be peer evaluation**
- **You are expected to use GitHub for this assignment, teamwork should be evident on GitHub**

Objectives

- **Practice using third party libraries**
- **Apply your knowledge about graphs for the solution of a real world problem**
- **Apply your knowledge about Time Complexity**
- **Reflect on the difficulties and how you surmounted them**

Description

There is a church where a community of people attend. This community studies the bible and to do so they form small groups to study specific topics to allow for in depth meditation and sharing between the members. One of the goals of the main community is that every member gets to know every other member, and that every one gets to go to everyone's home. The small group meetings happen in the house of the host.

Your goal is to design and develop a program that will read the list of people and the desired size of small groups.

There is one additional issue to consider: married couples always go together.

Example

John + Ann, Peter + Melissa, George, Lisa, Jenny, Alma, Jerry, James, Albert, Jack, Jill, Jaimie, July, Karl, Hector, Kathy, Bert, Homer, Jean, Joan, Geordi, Scott, Ivan, Yi-Ran, Abdul, Italo

Say, the small group size is 4.

The first iteration is easy

Example

John + Ann, Peter + Melissa, George, Lisa, Jenny, Alma, Jerry, James, Albert, Jack, Jill, Jaimie, July, Karl, Hector, Kathy, Bert, Homer, Jean, Joan, Geordi, Scott, Ivan, Yi-Ran, Abdul, Italo, Mark

G1,1: John + Ann, Peter + Melissa
G1,2: George, Lisa, Jenny, Alma
G1,3: Jerry, James, Albert, Jack
G1,4: Jill, Jaimie, July, Karl
G1,5: Hector, Kathy, Bert, Homer
G1,6: Jean, Joan, Geordi, Scott
G1,7: Ivan, Yi-Ran, Abdul, Italo, Mark

First Iteration

Note: since the list is not a multiple of 4, one person needs to be ”added” to the last group. If there was another person, then G1,6 and G1,7 would have an extra person

Example

John + Ann, Peter + Melissa, George, Lisa, Jenny, Alma, Jerry, James, Albert, Jack, Jill, Jaimie, July, Karl, Hector, Kathy, Bert, Homer, Jean, Joan, Geordi, Scott, Ivan, Yi-Ran, Abdul, Italo, Mark

G1,1: John + Ann, Peter + Melissa
G1,2: George, Lisa, Jenny, Alma
G1,3: Jerry, James, Albert, Jack
G1,4: Jill, Jaimie, July, Karl
G1,5: Hector, Kathy, Bert, Homer
G1,6: Jean, Joan, Geordi, Scott
G1,7: Ivan, Yi-Ran, Abdul, Italo, Mark

First Iteration

Note: since the list is not a multiple of 4, one person needs to be ”added” to the last group. If there was another person, then G1,6 and G1,7 would have an extra person.
The green name is the host

Example

John + Ann, Peter + Melissa, George, Lisa, Jenny, Alma, Jerry, James, Albert, Jack, Jill, Jaimie, July, Karl, Hector, Kathy, Bert, Homer, Jean, Joan, Geordi, Scott, Ivan, Yi-Ran, Abdul, Italo, Mark

G2,1: John + Ann, Jenny, Italo
G2,2: Peter + Melissa, Lisa, Alma
G2,3: Jerry, Jack , Karl, George,
G2,4: Jill, Joan , Albert , James
G2,5: Hector, Kathy, Bert, Homer
G2,6: Jean, Geordi, Scott , Jaimie
G2,7: Ivan, Yi-Ran, Abdul, Mark, July

Second Iteration

Note: since the list is not a multiple of 4, one person needs to be ”added” to the last group. If there was another person, then G1,6 and G1,7 would have an extra person.
The green name is the host

Program Output

The program will produce a list of lists. Each iteration produces a list of groups the most evenly distributed possible in such a way that there is a host for each group, married couples are always together and everyone gets to go to everybody else's home.

Something to think about

Given a list of n people with a size of small groups m, what is minimum number of iterations necessary to accomplish the goal of everybody visiting everybody's house?

What is the time complexity of such algorithm?

Deliverables

Three text files, one with 16 names, one with 29 names, one with 34 names. Include some married couples in all three files. Married couples should be in a single line separated by commas, then each name on its own line.

A README.md file with the instructions on how to run your program

The source code of your program

If your program is in Java, include the JAR file with the necessary lib files. If your program is in Python include the Requirements file.

A section inside the README.md with 500 words discussing: How to find the set of sets of teams in the minimum amount of iterations, the time complexity of that algorithm, the time complexity of your algorithm, and the difficulties you faced during this assignment and how you prevailed. Add screenshots of your program running, and add the set of sets for each file.

A link to a YouTube video showing your program running with an explanation of the steps that you are taking

Grading

PR	**Peer review form 0 – 10**
G	**The program uses a graph to model home visits 0/1**
V	**The video thoroughly shows the program running 0 - 15**
ALS	**The README file has all the sections: Introduction, Description, Requirements, User Manual, Reflection [0 – 5]**
Q[5]	**Quality of each section {[0 – 2], [0 – 2], [0 – 8], [0 – 8], [0 – 25]}**
I	**The instructions are enough to run the program and Does the program run? [0.0 – 1.0]**
R	**Requirements are clear and included (i.e. jar files for Java) [0 – 5]**
AN	**The program animates the graph as it is being built [0 – 15]**
QP	**Quality of the Program [0 – 20]**
PP	**Programming Practices [0 – 50]**

Grade = (PR + V + ALS + SUM(Q) + R + AN + QP – PP) * G * I

Sections to write on README

 # Title

Introduction

Description

Requirements

User Manual

Reflection

Files in your Repository

Three community names files:

- **group1.txt**
- **group2.txt**
- **group3.txt**

Your source code and files necessary to run it (JAR files in case of Java, Requirements in case of Python)