Student ID’s

Jeff Omidvaran 36128900   jomidvar@uci.edu

Shuo Liu 64563502 shuol22@uci.edu

Yun Lu 43206752 yunl36@uci.edu

Hanyan Wang 37961104 hanyanw3@uci.edu

https://github.com/jeffomidvaran/253P_Lab1

Identify edge cases for the problem:

1.groupSize># of students, each student has its own group

2.groupSize=0, cerr<<”no group”;

3. groupSize = 1, all students in one group

Define effective **test case**(s) and expected result(s) for program:

Roster.txt: (Valid Students Names Part)

A, B, C, D, E, F, G

Absent Student:

F,G

Expected results for different group size:

1. For group size=5:

[A], [B], [C], [D], [E]

2. For group size=0 :

[]

3. For group size=1:

[A, B, C, D, E]

4. For Groupsize 2:

[A,B,E], [C,D]

5. For Groupsize 3:

[A,D], [B,E], [C]

Design one **algorithmic solution** on paper (or whiteboard, or a text file.) put the screenshot or text here:

1. Store the absent student's name in an unordered set **absentStudents**.
2. Traversing the input file, and append students who are not absent into **students**.
3. Shuffle the **students**, then the names are randomly ordered.
4. Given the **groupSize**, we could calculate:
   a. # of students in each group **num**.
   b. # of left students **remain**.
5. Initialize **index,** and **forEach** group:
   a. Add **num** of students into it.
   b. Add a remaining student into it if **remain** larger than 0.

Analyze the **time** and space **complexity** of the solution:

Let `n` be the total number of students. `k` be the number of absent students. Note k<n always.

1. Time: O(k). Space: O(n).
2. Time: O(n). Space: O(n-k).
3. Time: O(n-k). Space: O(1).
4. Time: O(1). Space: O(1).
5. Time: O(n). Space: O(1).

Write nearly correct **code on paper** (or whiteboard, or text file) to solve problem:

Python: (Clean for demonstrating our idea)

```
def lab1(students, group_size, absent_students):

    absent = set(absent_students)

    stud = []

     for name in students:

         if name not in absent:

             stud.append(name)

     if group_size>len(stud) or not group_size:

         return None

     shuffle(stud)

    res = []

    k, left = len(stud)//group_size, len(stud)%group_size

    i = 0

    for group in range(group_size):

        res.append(stud[i:i+k])

        i += k

    for j in range(left):

        res[j].append(stud[i])

        i += 1

     return res
```

C++:

```
#include
<iostream>

#include <fstream>

#include <streambuf>

#include <vector>

#include <optional>

#include <unordered_set>

#include <algorithm>

#include <string>

#include <boost/algorithm/string.hpp>

#include <unordered_set>



using namespace std;



vector<string> students;

// vector<string> absentStudents;

unordered_set<string> absentStudents;



string getFileText() {
   ifstream myFile("roster.txt");

   string str((istreambuf_iterator<char>(myFile)),
istreambuf_iterator<char>());

   myFile.close();

   return str;

}




int getPerson(int startIndex, string fileText, int endIndex) {

   int startIndexLastName = fileText.find("\t", startIndex) + 1;

   if (startIndexLastName > endIndex) {

       return -1;

   }

   int endIndexLastName = fileText.find(",", startIndexLastName);

   string lastName = fileText.substr(startIndexLastName,
endIndexLastName - startIndexLastName);



   int startIndexFirstName = endIndexLastName + 2;

   int endIndexFirstName = fileText.find("\t", startIndexFirstName);

   string firstName = fileText.substr(startIndexFirstName,
endIndexFirstName - startIndexFirstName);



   // take out absent students before they are added in.

   string currentStudent = firstName + " " + lastName;
   unordered_set<string>::const_iterator found  =
absentStudents.find(currentStudent);

   if (found != absentStudents.end()){

       return fileText.find("\n", endIndexLastName);

   }




   students.push_back(currentStudent);

   return fileText.find("\n", endIndexLastName);

}









void getAbsentStudents() {

   string absentStudent;

   do {

       cout << "Enter the name of an absent Student (Type \"end\" to
stop entering absent students): ";

       getline(cin, absentStudent);

       if(absentStudent != "end") {

           absentStudent = boost::to_upper_copy(absentStudent);

           absentStudents.insert(absentStudent);

       }

   }while(absentStudent != "end");
```

```
}



void getStudentsFromFileText(string fileText) {

   string studentDescriptionHeader = "Student# Name   Email  Major
Lvl    Opt    Notes";

   int startIndex = fileText.find(studentDescriptionHeader) +
studentDescriptionHeader.length() + 2;

   int endIndex =  fileText.find("Total: ");



   do {

       startIndex = getPerson(startIndex, fileText, endIndex);

   } while(startIndex != -1);

}



void createGroups(int size) {

   random_shuffle(students.begin(),students.end());

   int num=students.size()/size;

   int remain=students.size()%size;

   cout<<endl<<"groups:"<<endl;

   int index=0;

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

       cout<<i<<". \n";

       for(int j=0;j<num;j++){
```

```
                cout<< "\t" << students[index] << endl;

                index++;

            }

            if(remain>0){

                cout<< "\t" << students[index] << endl;

                // cout<<students[index]<<endl;

                index++;

                remain--;

            }

        }

}





int main(int argc, char** argv) {

    /*

        group size comes in from stdin argv[1]

        absent students is entered by the user via command line

    */

    getAbsentStudents();

    string fileText = getFileText();

    getStudentsFromFileText(fileText);

    createGroups(stoi(argv[1]));
    return 0;
}
```

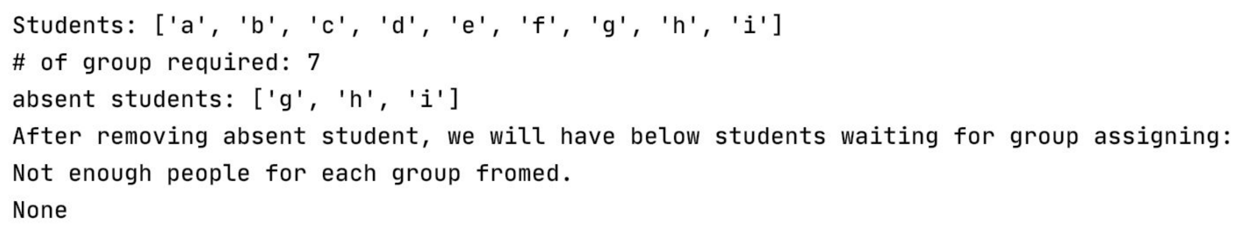
Explain your algorithm/program in simple words:

We first removed absent students from the students list we read from the file, since they won’t be assigned to any group. Then we shuffled the student list, and students would be randomly ordered.

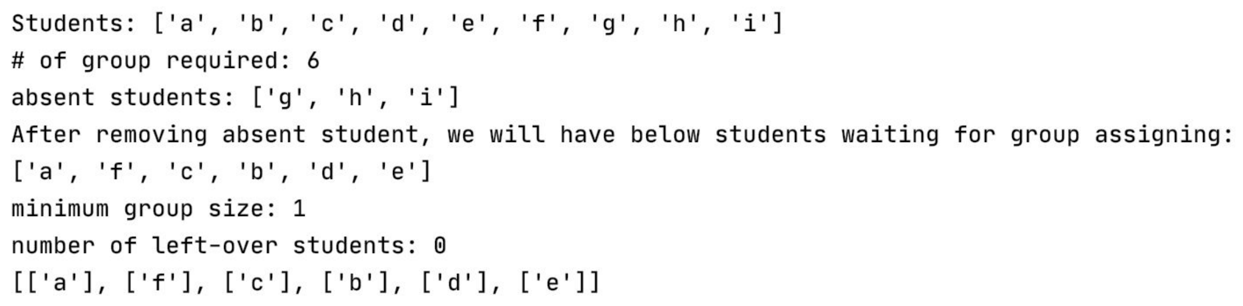
To create the groups, we first calculated the minimum number of students **num** in each group and the number of left-over students **remain**. For each group, we go **num** steps and add a student if there exists remaining students, saying **remain** > 0. Finally, the students are grouped.

Simulate test case and verify your program produces correct results:

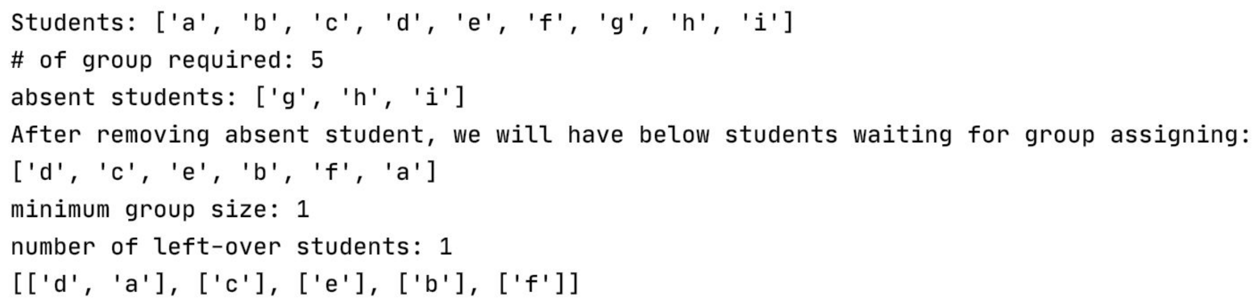
Python results:



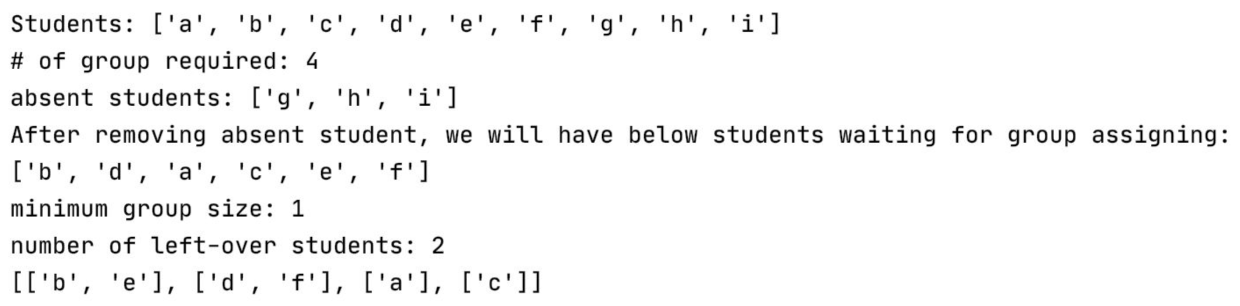
```
Students: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
# of group required: 7
absent students: ['g', 'h', 'i']
After removing absent student, we will have below students waiting for group assigning:
Not enough people for each group fromed.
None
```



```
Students: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
# of group required: 6
absent students: ['g', 'h', 'i']
After removing absent student, we will have below students waiting for group assigning:
['a', 'f', 'c', 'b', 'd', 'e']
minimum group size: 1
number of left-over students: 0
[['a'], ['f'], ['c'], ['b'], ['d'], ['e']]
```



```
Students: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
# of group required: 5
absent students: ['g', 'h', 'i']
After removing absent student, we will have below students waiting for group assigning:
['d', 'c', 'e', 'b', 'f', 'a']
minimum group size: 1
number of left-over students: 1
[['d', 'a'], ['c'], ['e'], ['b'], ['f']]
```

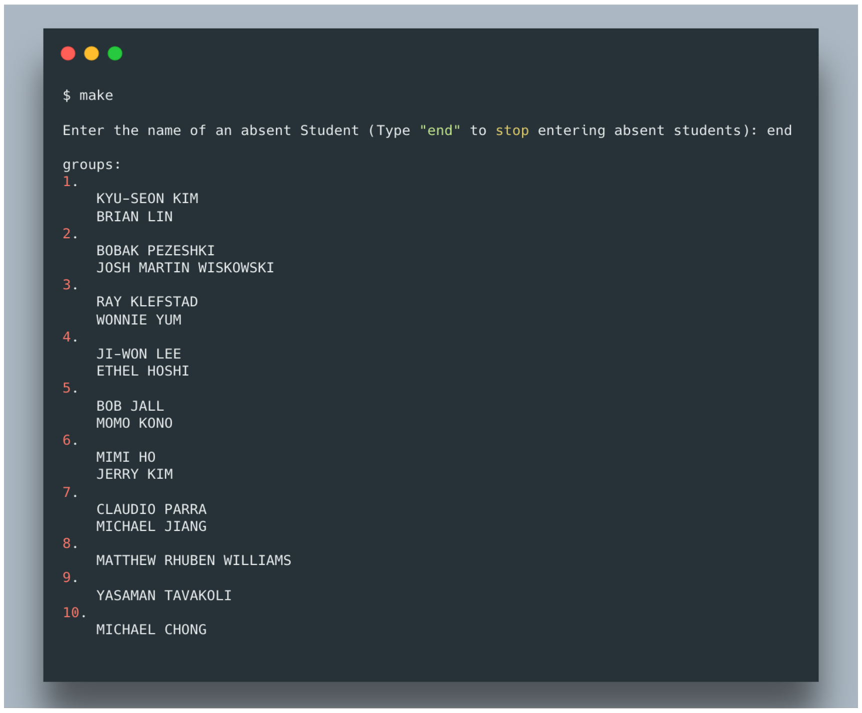


```
Students: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
# of group required: 4
absent students: ['g', 'h', 'i']
After removing absent student, we will have below students waiting for group assigning:
['b', 'd', 'a', 'c', 'e', 'f']
minimum group size: 1
number of left-over students: 2
[['b', 'e'], ['d', 'f'], ['a'], ['c']]
```

```
Students: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
# of group required: 3
absent students: ['g', 'h', 'i']
After removing absent student, we will have below students waiting for group assigning:
['e', 'd', 'f', 'c', 'b', 'a']
minimum group size: 2
number of left-over students: 0
[['e', 'd'], ['f', 'c'], ['b', 'a']]
```

C++ results:

10 groups with no absent:

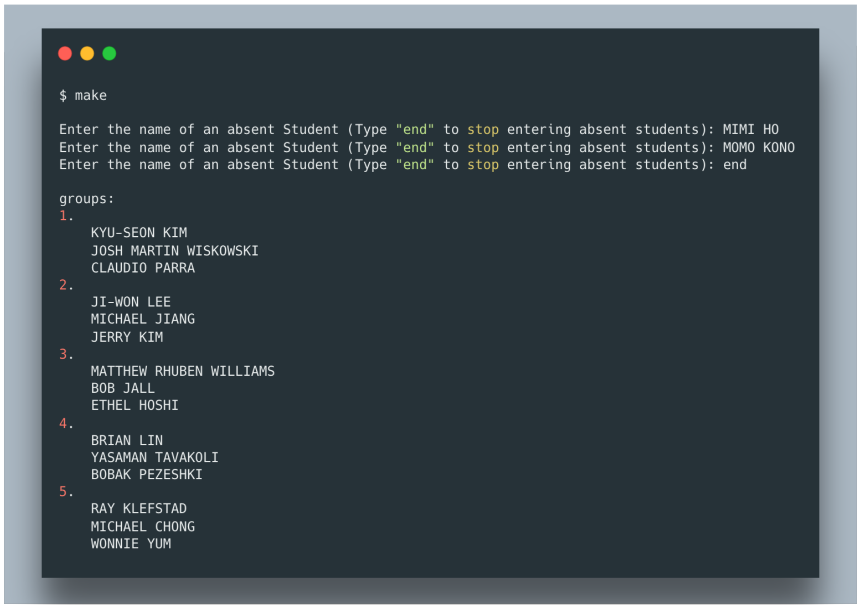


```
$ make

Enter the name of an absent Student (Type "end" to stop entering absent students): end

groups:
1.
    KYU-SEON KIM
    BRIAN LIN
2.
    BOBAK PEZESHKI
    JOSH MARTIN WISKOWSKI
3.
    RAY KLEFSTAD
    WONNIE YUM
4.
    JI-WON LEE
    ETHEL HOSHI
5.
    BOB JALL
    MOMO KONO
6.
    MIMI HO
    JERRY KIM
7.
    CLAUDIO PARRA
    MICHAEL JIANG
8.
    MATTHEW RHUBEN WILLIAMS
9.
    YASAMAN TAVAKOLI
10.
    MICHAEL CHONG
```

5 groups with 2 absent:

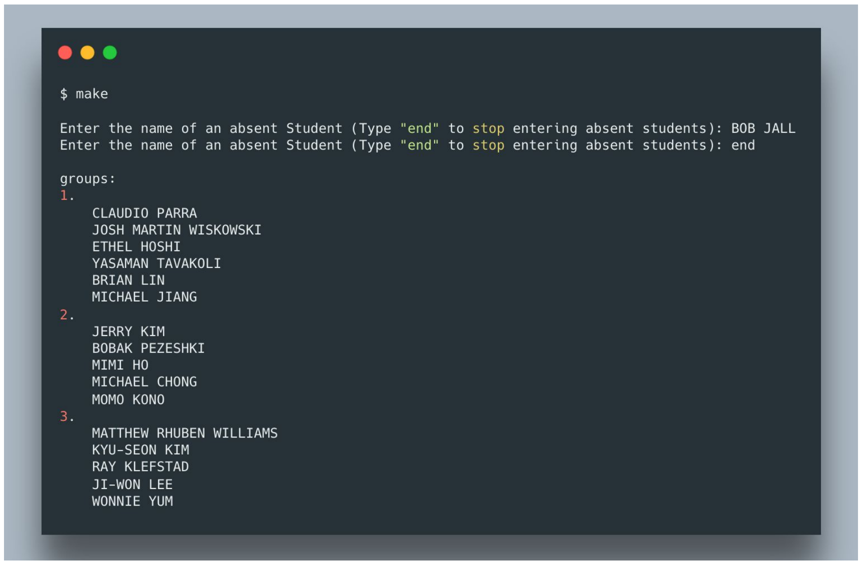


```
$ make

Enter the name of an absent Student (Type "end" to stop entering absent students): MIMI HO
Enter the name of an absent Student (Type "end" to stop entering absent students): MOMO KONO
Enter the name of an absent Student (Type "end" to stop entering absent students): end

groups:
1.
    KYU-SEON KIM
    JOSH MARTIN WISKOWSKI
    CLAUDIO PARRA
2.
    JI-WON LEE
    MICHAEL JIANG
    JERRY KIM
3.
    MATTHEW RHUBEN WILLIAMS
    BOB JALL
    ETHEL HOSHI
4.
    BRIAN LIN
    YASAMAN TAVAKOLI
    BOBAK PEZESHKI
5.
    RAY KLEFSTAD
    MICHAEL CHONG
    WONNIE YUM
```

3 groups with one absent:



```
$ make

Enter the name of an absent Student (Type "end" to stop entering absent students): BOB JALL
Enter the name of an absent Student (Type "end" to stop entering absent students): end

groups:
1.
    CLAUDIO PARRA
    JOSH MARTIN WISKOWSKI
    ETHEL HOSHI
    YASAMAN TAVAKOLI
    BRIAN LIN
    MICHAEL JIANG
2.
    JERRY KIM
    BOBAK PEZESHKI
    MIMI HO
    MICHAEL CHONG
    MOMO KONO
3.
    MATTHEW RHUBEN WILLIAMS
    KYU-SEON KIM
    RAY KLEFSTAD
    JI-WON LEE
    WONNIE YUM
```