Jacob Alspaw
EECS 338
Assignmnet 3

Variables:

Binary Semaphores

- mutex, initialized to 1 because no process will start using shared resources
- tina, initialized to 0 because they start off in a sleep state / blocked from cutting hair
- judy, initialized to 0 because they start off in a sleep state / blocked form cutting hair

List

- a single line where every customer waits regardless of worker preference

Booleans

- isTinaBusy, will denote if Tina is busy. Sleeping is not busy. Initialized to false.
- isJudyBusy, will denote is Judy is busy. Sleeping is not busy. Initialized to false.

Integers

- potentialTinaLineLength, initialized to 0 because no line at start
- potentialJudyLineLength, initialized to 0 because no line at start

Assumptions:

- Everyone is standing in a single line.
- No preference customers count as a potential customer for Judy and for Tina, and will therefore be included in any customer type's consideration for leaving the store. A Tina customer will not leave if Judy has a line of customers that exceeds five, and vice versa.

Tina:

```
while(true) {
        wait(tina);
        wait(mutex);
        cutsomerToCut = takeNextPotentialCustomerInLineForTina();
        potentialTinaLineLength--;
        if(customerToCut.type == NoPreference) {
                potentialJudyLineLength--;
        }
        signal(mutex);
        cutHair(customerToCut);
        wait(mutex);
        isTinaBusy = false;
        if(potentialTinaLineLength > 0) {
                isTinaBusy = true;
                signal(tina);
        }
        signal(mutex);
}
```

Explanation: Tina is always cutting hair. We wait for Tina to be needed. We wait for access to mutually exclusive variables (the line). We find, remove, and remember the next customer in the single line that will let Tina cut their hair. We decrement number of people in line that will let Tina cut their hair. If that person didn’t care who cut their hair, then we must also decrease the amount of people that would let Judy cut their hair. We stop accessing mutually exclusive variables (the line). We cut the persons hair. We wait for access to shared variables and Tina is no longer busy because she has finished cutting that person’s hair. She checks the line to see if anyone else needs their hair cut. If someone is in line that needs their hair cut, Tina will tell herself to repeat the process above after ensuring mutual exclusivity. Otherwise, Tina will go back to sleeping.

Judy:

```
while(true) {
    wait(judy);
    wait(mutex);
    cutsomerToCut = takeNextPotentialCustomerInLineForJudy();
    potentialJudyLineLength--;
    if(customerToCut.type == NoPreference) {
        potentialTinaLineLength--;
    }
    signal(mutex);
    cutHair(customerToCut);
    wait(mutex);
    isJudyBusy = false;
    if(potentialJudyLineLength > 0) {
        isJudyBusy = true;
        signal(judy);
    }
    signal(mutex);
}
```

Explanation: Judy is always cutting hair. We wait for Judy to be needed. We wait for access to mutually exclusive variables (the line). We find, remove, and remember the next customer in the single line that will let Judy cut their hair. We decrement number of people in line that will let Judy cut their hair. If that person didn't care who cut their hair, then we must also decrease the amount of people that would let Tina cut their hair. We stop accessing mutually exclusive variables (the line). We cut the persons hair. We wait for access to shared variables and Judy is no longer busy because she has finished cutting that person's hair. She checks the line to see if anyone else needs their hair cut. If someone is in line that needs their hair cut, Judy will tell herself to repeat the process above after ensuring mutual exclusivity. Otherwise, Judy will go back to sleeping.

Tina-Only:

```
wait(mutex);
if(potentialTinaLineLength < 5) {
        customer.addToLine();
        potentialTinaLineLength++;
        if(isTinaBusy = = false) {
                isTinaBusy = true;
                signal(tina);
        }
}
else {
        customer.leave();
}
signal(mutex);
```

Explanation: Tina only customers will enter the store and wait to access shared variables. The customer will check if the number of customers already in line that will let Tina cut their hair exceeds 5. If so, then the customer will leave the store. If the worst-case wait time is 5 or less people, then the customer will add themselves to the back of the line. If Tina is not busy (meaning she is sleeping), then the customer will notify Tina of his presence. The customer will stop access to shared variables.

Judy-Only:

```
wait(mutex);
if(potentialJudyLineLength < 5) {
    customer.addToLine();
    potentialJudyLineLength++;
    if(isJudyBusy = = false) {
        isJudyBusy = true;
        signal(judy);
    }
}
else {
    customer.leave();
}
signal(mutex);
```

Explanation: Judy only customers will enter the store and wait to access shared variables. The customer will check if the number of customers already in line that will let Judy cut their hair exceeds 5. If so, then the customer will leave the store. If the worst-case wait time is 5 or less people, then the customer will add themselves to the back of the line. If Judy is not busy (meaning she is sleeping), then the customer will notify Judy of his presence. The customer will stop access to shared variables.

No Preference:

```
wait(mutex);
if(line.length < 5) {
          customer.addToLine();
          potentialTinaLineLength++;
          potentialJudyLineLength++;
          if(isTinaBusy = = false) {
                    isTinaBusy = true;
                    signal(tina);
          }
          else if(isJudyBusy = = false) {
                    isJudyBusy = true;
                    signal(judy);
          }

}
else {
          customer.leave();
}
signal(mutex);
```

Explanation: No preference customers will enter the store and wait to access shared variables. The customer will check if the number of customers already in line exceeds 5. If so, then the customer will leave the store. If there are 5 or less people waiting to get their haircuts, then the customer will add themselves to the back of the line. If Tina is not busy (meaning she is sleeping), then the customer will notify Tina of his presence. He will check with Tina first, because Tina owns the store. If Tina is busy, but Judy is not, then the customer will notify Judy of his presence. The customer will stop access to shared variables.