# Assignment 2

- **Initialize the Bank Simulation:**
    - Set up the number of tellers, a queue for customers, and hash tables for tracking tellers' status, idle times, customers served, and least busy times.
    - Each teller starts idle with no customer assigned.
- **Read and Parse Input**
    - Read customer data from the input file. Each customer has an arrival time, service time, and priority. Customers are added to a list for processing.
- **Run the Simulation:**
    - Iterate over all customers based on their arrival times.
- **For each customer:**
    - Update the simulation time to match the customer's arrival time.
    - Try to allocate the customer to an idle teller using a **round-robin** approach.
    - If no tellers are idle, place the customer in a **priority queue** (MinHeap) based on arrival time and priority.
    - Update teller status and the queue as tellers finish serving customers.
- After processing all customers, continue simulating to serve the remaining customers in the queue.
- **Allocate Customers to Tellers:**
    - Assign a customer to the first available idle teller (checked cyclically via round-robin).
    - If no tellers are idle, add the customer to the queue.
- **Update Teller Status:**
    - Regularly check the status of each teller. If a teller has finished serving a customer:
    - Mark the teller as idle and update the teller's last busy time.
    - If the queue is not empty, assign the next customer to the teller.
- **Track and Compute Statistics:**
    - Track the number of customers served by each teller.
    - Calculate teller idle times, queue length, and total simulation time.
    - Compute metrics such as average service time per customer, average queue length, and average waiting time per customer.
- **Print Simulation Results:**
    - Display key statistics after the simulation finishes:
        - Number of customers served by each teller.
        - Total simulation time.
        - Average service and waiting times.
        - Maximum and average queue lengths.
        - Teller idle rates.

Student#: 81167765
Name: Michael McMillan

**1. HashTable Operations:**

- **Insert:**
  In the hash table, inserting a key-value pair involves computing the hash, probing for an empty slot (if necessary), and inserting the item.
  **Worst-case time complexity:**
  In the case of collisions, probing may continue until we find an empty slot, leading to O(n) time, where n is the size of the hash table.
  - **Average-case time complexity:** O(1), assuming a low load factor.
  - **Worst-case time complexity:** O(n) due to probing in case of hash collisions.
- **Get/Update:**
  Both getting and updating require computing the hash and possibly probing in case of collisions.
  - **Average-case time complexity:** O(1).
  - **Worst-case time complexity:** O(n) in the case of many collisions and clustering in the hash table.
- **Space complexity:**
  The hash table requires space proportional to its size and the number of stored key-value pairs.
  - **Space complexity:** O(n), where n is the number of key-value pairs stored.

---

**2. MinHeap Operations (Queue):**

- **Insert:**
  Inserting into a min-heap involves adding the new element at the end and "heapifying up" to restore the heap property.
  - **Time complexity:** O(log k), where k is the number of elements in the heap at the time of insertion.
- **Extract Min (Dequeue):**
  Removing the minimum element from a min-heap involves replacing the root with the last element, removing the last element, and "typifying down" to restore the heap property.
  - **Time complexity:** O(log k), where k is the number of elements in the heap.
- **Space complexity:**
  The space needed for the heap is proportional to the number of customers in the queue.
  - **Space complexity:** O(n), where n is the number of customers.

---

**3. BankSimulation Methods:**

Student#: 81167765
Name: Michael McMillan

- **read_file:**
  Reading and parsing the input file to generate customer objects involves processing each line and extracting data for each customer.
    - **Time complexity:** O(m), where m is the number of lines (or customers) in the input file.
    - **Space complexity:** O(m) for storing the customer objects.
- **run:**
  This method processes all customers and runs the simulation:
    - For each customer:
        - Allocating a customer involves a **HashTable update** or an **enqueue operation** in the **MinHeap**.
        - **Update tellers**: Involves iterating through num_tellers to check their status.
    - After all customers are processed, the method continues until the queue is empty, performing dequeue operations from the **MinHeap** and updating tellers.
    - **Time complexity per customer:**
        - Allocating a customer to a teller: O(1) on average for HashTable update, worst case O(num_tellers) in probing.
        - If enqueuing in **MinHeap**, O(log k), where k is the number of customers in the queue.
        - Updating the status of num_tellers: O(num_tellers).
    - **Overall time complexity for a run:**
        - O(m log m + m * num_tellers), where m is the number of customers and num_tellers is the number of tellers.
- **allocate_customer:**
  The allocation process involves checking the round-robin status, updating a teller, or enqueuing the customer into the MinHeap.
    - **Time complexity:** O(1) for HashTable updates, or O(log m) for MinHeap insertion.
- **update_tellers:**
  This involves iterating over all tellers and updating their status. If the teller has finished serving, it also dequeues from the queue (which involves extracting the minimum from the MinHeap).
    - **Time complexity:** O(num_tellers) for iterating over all tellers.
    - **If dequeuing a customer:** O(log m) for MinHeap extraction.
- **find_idle_teller:**
  This method iterates over all tellers in a round-robin manner to find an idle teller.
    - **Time complexity:** O(num_tellers), where num_tellers is the number of tellers.

---

Student#: 81167765
Name: Michael McMillan

**Overall Time Complexity:**

- **Initial Setup:**
  - read_file: O(m) to process m customers.
- **Simulation Run:**
  - For each customer, we perform:
    - Hash table operations (insertion/update), each costing O(1) on average.
    - Enqueueing into the min-heap, which costs O(log m).
    - Teller updates (checking all num_tellers), costing O(num_tellers).
  - For all m customers: O(m log m + m * num_tellers).
- **Post-customer processing (emptying queue):**
  - The queue may contain all remaining customers, leading to an additional O(m log m) complexity for dequeuing all customers from the heap.
- **Total time complexity:**
  O(m log m + m * num_tellers), where:
  - m is the number of customers.
  - num_tellers is the number of tellers.

---

**Overall Space Complexity:**

- **Customers list:** O(m) for storing m customers.
- **HashTables:** O(num_tellers) for storing teller statuses, idle times, etc.
- **MinHeap (Queue):** O(m) for storing customers in the queue.
- **Total space complexity:**
  O(m + num_tellers), where:
  - m is the number of customers.
  - num_tellers is the number of tellers.

---

**Summary:**

- **Time complexity:** O(m log m + m * num_tellers)
- **Space complexity:** O(m + num_tellers)

The simulation is efficient as long as the number of tellers (num_tellers) remains reasonably small relative to the number of customers (m).

Student#: 81167765
Name: Michael McMillan

## Data Structures and reasons for using them

1. **HashTable:**
    - **Purpose:** Used for storing and managing various data types efficiently, such as teller statuses, idle times, customers served, and least busy times.
    - **Reasons for Use:**
        - **Fast Access:** Provides average O(1) time complexity for insertion, lookup, and update operations, which is essential for maintaining and accessing teller statuses and statistics quickly.
        - **Key-Value Mapping:** Allows for the association of teller IDs with their respective statuses and metrics, making it easy to retrieve and update information related to each teller.
2. **MinHeap:**
    - **Purpose:** Used to manage a priority queue of customers, where customers are prioritised based on their arrival time and service time.
    - **Reasons for Use:**
        - **Efficient Priority Management:** MinHeap supports efficient extraction of the customer with the highest priority (i.e., the one who has been waiting the longest or has the highest priority) in O(log k) time, where k is the number of customers in the heap.
        - **Dynamic Insertions and Removals:** This feature enables efficient insertion and removal of customers from the queue, ensuring that the customers with the most urgent needs are served first.
3. **List:**
    - **Purpose:** Store the customer's data from the input file.
    - **Reasons for Use:**
        - **Ordered Storage:** This maintains customers' orders as they are read from the file, which is crucial for processing them based on their arrival times.
        - **Ease of Use:** Provides simple operations for iterating and accessing customer data.
4. **Tuple:**
    - **Purpose:** Used within the hash table to store a pair of values: the customer object and the finish time for that customer.
    - **Reasons for Use:**
        - **Immutable Data Structure:** Tuples are immutable, which means their values cannot be changed once they are created. This is useful for ensuring that the key-value pair (customer, finish time) remains consistent until the customer is served.
5. **Primitive variables**

Student#: 81167765
Name: Michael McMillan

- **Purpose:** Variables like `time`, `max_queue_length`, `total_queue_time`, and `round_robin_counter` are used to track simulation time, maximum queue length, total time customers spend in the queue and the current position in the round-robin process.
- **Reasons for Use:**
  - **Tracking and Calculation:** This is essential for managing the simulation's state and calculating various metrics, such as the total queue length, average service time, and teller idle times.

## Snapshot of the execution of the program

## Outputs produced by the program

### 1 teller for simulation

```
C:\Users\michael\anaconda3\envs\CSCI203\python.exe "C:\Users\michael\OneDrive - University of Wollongong\UOW\Bachelor of Computer Science\Year2\Spring\CSCI203 - Algorithms and Data Structures\Assessments\Assignment 2\main.py"
Please enter the number of tellers: 1
Please enter the name of the input file: a2-sample.txt
Simulation Inputs
Number of tellers: 1
Name of input file: a2-sample.txt

Simulation Statistics
Customers Served by Each Teller
Teller [0]: 100
Total Time of Simulation: 1416.48
Average Service Time per Customer: 9.7773
Average Waiting Time per Customer: 913.412
Maximum Length of the Queue: 65
Average Length of the Queue: 2.2944
Average Idle Rate of Each Teller
Teller [0]: 0.9992940

Process finished with exit code 0
```

### 2 tellers for simulation

```
C:\Users\michael\anaconda3\envs\CSCI203\python.exe "C:\Users\michael\OneDrive - University of Wollongong\UOW\Bachelor of Computer Science\Year2\Spring\CSCI203 - Algorithms and Data Structures\Assessments\Assignment 2\main.py"
Please enter the number of tellers: 2
Please enter the name of the input file: a2-sample.txt
Simulation Inputs
Number of tellers: 2
Name of input file: a2-sample.txt

Simulation Statistics
Customers Served by Each Teller
Teller [0]: 51
Teller [1]: 49
Total Time of Simulation: 750.48
Average Service Time per Customer: 12.3562
Average Waiting Time per Customer: 250.853
Maximum Length of the Queue: 35
Average Length of the Queue: 2.1786
Average Idle Rate of Each Teller
Teller [0]: 0.9866752
Teller [1]: 0.9986675

Process finished with exit code 0
```

Student#: 81167765
Name: Michael McMillan

## 4 tellers for simulation

```
C:\Users\michael\anaconda3\envs\CSCI203\python.exe "C:\Users\michael\OneDrive - University of Wollongong\UOW\Bachelor of Computer Science\Year2\Spring\CSCI203 - Algorithms and Data Structures\Assessments\Assignment 2\main.py"
Please enter the number of tellers: 4
Please enter the name of the input file: a2-sample.txt
Simulation Inputs
Number of tellers: 4
Name of input file: a2-sample.txt

Simulation Statistics
Customers Served by Each Teller
Teller [0]: 28
Teller [1]: 25
Teller [2]: 24
Teller [3]: 23
Total Time of Simulation: 509.48
Average Service Time per Customer: 15.7294
Average Waiting Time per Customer: 0.000
Maximum Length of the Queue: 2
Average Length of the Queue: 0.0981
Average Idle Rate of Each Teller
Teller [0]: 0.9854510
Teller [1]: 0.9854510
Teller [2]: 1.0000000
Teller [3]: 0.9854510

Process finished with exit code 0
```

Student#: 81167765
Name: Michael McMillan

# Discussion on the Effect of Teller Number on Service Efficiency

1. **Customers Served by Each Teller:**
    - **Teller [0]:** 28 customers
    - **Teller [1]:** 25 customers
    - **Teller [2]:** 24 customers
    - **Teller [3]:** 23 customers

**Observation:** The number of customers served is relatively balanced among the tellers, with a slight variation. Teller [0] served the most customers, while Teller [3] served the least. This variation is likely due to the round-robin scheduling approach, which ensures that each teller gets a chance to serve customers but may not be perfectly balanced due to the randomness of customer arrivals and service times.

2. **Total Time of Simulation:**
    - **Total Time:** 509.48 seconds

**Observation:** The total simulation time represents the period from the start until the empty queue. This time is a direct result of customer arrivals, service times and the efficiency of teller management.

3. **Average Service Time per Customer**
    - **Average Service Time:** 15.7294 seconds

**Observation:** This is the average time each customer spends being served. With 4 tellers, this time is relatively low, indicating that customers are being served quickly.

4. **Average Waiting Time per Customer**
    - **Average Waiting Time:** 0 seconds

**Observation:** This is the average time each customer waits in the queue before being served. A higher waiting time is expected in simulations with more complex queries or longer service times, but this value provides insight into how well the system handles customer traffic. Since the average waiting time is 0 seconds, this suggests that each teller is constantly serving the customers.

5. **Maximum Length of the Queue**
    - **Maximum Length:** 2

**Observation:** The maximum queue length is 2, indicating that up to 2 customers were waiting to be served at peak times. This relatively small queue length suggests that the system handles customer flow efficiently.

6. **Average Length of the Queue**
    - **Average Length:** 0.0981

**Observation:** The average length of the queue is very low, which suggests that the queue is rarely congested. This implies that the number of tellers can handle customer arrivals without causing significant delays.

Student#: 81167765
Name: Michael McMillan

7. **Average Idle Rate of Each Teller**
    - Teller [0]: 0.9854510
    - Teller [1]: 0.985410
    - Teller [2]: 1.000000
    - Teller [3]: 0.9854510

**Observation:** The idle rate measures how often each teller does not serve customers. Teller [2] has the highest idle rate, indicating it was idle more frequently than the others. The differences in idle rates suggest that while the tellers are generally busy, some might have more downtime, possibly due to their specific assignment in the round-robin scheduling.

## Effect of Number of Tellers on Efficiency

1. **Increased Teller Efficiency:**
    - **Queue Lengths:** With more tellers, the queue length is minimal, reducing customer wait times and improving overall service efficiency.
    - **Service Time:** More tellers generally lead to a lower average service time per customer, as each teller handles a portion of the customer load.
2. **Idle Time:**
    - **Teller Utilisation:** A higher number of tellers can lead to increased idle times if there aren't enough customers to keep all tellers busy. In your simulation, some tellers have a higher idle rate, indicating that while the number of tellers helps reduce queue lengths, not all tellers are fully utilised all the time.
3. **Balancing Teller Numbers:**
    - **Optimal Number:** The key to efficiency is finding the optimal number of tellers where customer wait times are minimised and teller utilisation is maximised. Too few tellers lead to longer queues and higher wait times, while too many tellers can lead to underutilisation and higher idle rates.

Student#: 81167765
Name: Michael McMillan